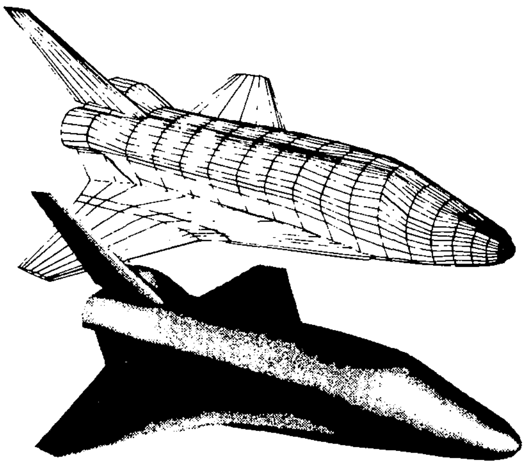
NASA Conference Publication 2272

Computer-Aided Geometry Modeling



*Abstracts of papers presented
at a symposium held at
NASA Langley Research Center
Hampton, Virginia
April 20-22, 1983*

NASA

NASA Conference Publi

Computer-Aided Geometry Modeling

Compiled by
John N. Shoosmith
and Robert E. Fulton
NASA Langley Research Center
Hampton, Virginia

Abstracts of papers presented
at a symposium held at
NASA Langley Research Center
Hampton, Virginia
April 20-22, 1983

NASA

National Aeronautics
and Space Administration

Scientific and Technical
Information Branch

1984

PREFACE

This publication contains abstracts and related figures of papers presented at the Symposium on Computer-Aided Geometry Modeling, held at NASA Langley Research Center, on April 20-22, 1983.

The continued growth in computers of various sizes, capabilities, and connectivities is providing increasingly powerful tools for solutions to scientific and engineering analysis, design, and manufacturing problems having three-dimensional geometries. The technology for computer-aided geometry modeling to support such work has reached the stage where numerous commercial systems now embody "wire frame" three-dimensional geometric descriptions and such systems are now widely used in a broad spectrum of applications. The next steps needed to advance technology for computer-aided geometry modeling include improvement in mathematical modeling, extension to solid geometry models, management of geometric data, development of appropriate geometry standards, and improvement in user interfaces through interactive and graphic procedures. This Symposium provides both invited and submitted papers in such areas, as well as in applications to several representative problem areas. The goal of this conference is to provide a focal point to better understand the state-of-the-art and to assess future research trends in computer-aided geometry modeling.

This Symposium was conceived through discussions with a variety of experts in the field including participants at a similar conference at the University of Utah in 1974, participants at the grid generation conference sponsored by NASA and the U.S. Air Force in April 1982, and staff at the Institute for Computer Applications in Science and Engineering (ICASE). Impetus at the Langley Research Center for computer-aided geometry modeling comes from two areas: (1) computer-aided design, where a major aspect of the IPAD (Integrated Programs for Aerospace-Vehicle Design) research is to represent the geometry of aircraft and space vehicles and to manage the data associated with evolving designs and (2) the solution to three-dimensional Navier-Stokes fluid flow equations for problems typified by the wing fuselage juncture. Both problem areas use three-dimensional geometry of complex objects as the starting point for discrete solutions on the computer. Financial support for the Symposium was provided by the NASA Office of Aeronautics and Space Technology, Aerospace Research Division, Fluid and Thermal Physics Office.

The Symposium was organized into eight topical areas, each having an invited paper and a small number of contributed papers. Abstracts of the papers are presented in these proceedings in the order in which they were given. In addition, a poster session was scheduled on the first afternoon, and abstracts for the poster papers are grouped together in the proceedings under Session IV.

This volume was prepared for publication through the efforts of Susan W. Bostic, Symposium Coordinator, and the staff of the Research Information and Applications Division, NASA Langley Research Center. Use of trade names or manufacturers' names in this publication does not constitute endorsement, either expressed or implied, by the National Aeronautics and Space Administration.

John N. Shoosmith, Symposium Co-Chairman
Robert E. Fulton, Symposium Co-Chairman

CONTENTS

SESSION V - ROLE OF GRAPHICS
IN GEOMETRY MODELING

SESSION VI - INTERACTIVE MODELING

SESSION VII - MANAGEMENT OF GEOMETRIC DATA

SESSION VIII - STATUS OF EVOLVING NATIONAL STANDARDS

SESSION IX - APPLICATIONS OF
GEOMETRY MODELING

SESSION X - APPLICATION OF
COMMERCIAL SYSTEMS

AN OPERATOR CALCULUS FOR SURFACE AND VOLUME MODELING*

William J. Gordon
Mathematical Sciences Department
Drexel University
Philadelphia, Pennsylvania

I. An Operator Calculus for Bivariate and Multivariate Approximation

The purpose of this paper is to briefly describe the mathematical techniques which form the foundation for most of the surface and volume modeling techniques used in practice. The first part is concerned with an outline of what may be termed an *operator calculus* for the approximation and interpolation of functions of more than one independent variable. By considering the linear operators associated with bivariate and multivariate interpolation/approximation schemes, we show how they can be compounded by operator multiplication and Boolean addition to obtain a distributive lattice of approximation operators. In the next two sections we show via specific examples how this operator calculus leads to practical techniques for sculptured surface and volume modeling.

To begin, let P denote a *projector* (i.e., an idempotent linear operator) which transforms functions F from some linear space Ψ into a subspace Φ. We may, for example, think of P as an approximation or interpolation operator which maps F into an approximation thereof. More specifically, one may think of P as the operator associated with simple linear interpolation to the end-point values of a univariate function F(x) on the interval [0, 1]: $P[F]=(1-x)F(0) + x\,F(1)$. In general, we wish to think of the linear space F as a space of functions of n independent variables so that F is properly denoted as $F(x_1, x_2, \ldots, x_n)$. Then Ψ may, for instance, be thought of as the linear space of all continuous functions in the independent variables $x_1, \ldots, x_n$.

Clearly, any function $F \in \Psi$ can be decomposed as follows: $F=P[F] + (I-P)[F]$ wherein I is the identity operator. If we think of $P[F]$ as an approximation to F, then $R=I-P$ is obviously the associated remainder operator. If $P[F]$ is a "good" approximation to F, then R[F] will be, in some sense, "small".

We wish now to consider a collection of projectors $P_1, P_2, \ldots, P_m$, all of which are defined on functions $F \in \Psi$. The result, $P_i[F]$, of applying P_i to any function in Ψ is a function in a subspace Φ_i; i.e., Φ_i is the range of the projector P_i. Since $P_i(P_i[F])=P_i[F]$ by virtue of the idempotency property of the operator P_i, Φ_i is also termed the exactness set of the *invariance set* of P_i. Although there are practically useful projectors which do not satisfy the following assumption, we will postulate that the collection of projectors is such that they are pairwise *commutative:* $P_iP_j[F]=P_jP_i[F]$, for any i and j.

The product of any two commutative projectors is also a projector since it is both linear and idempotent: $(P_iP_j)(P_iP_j)=P_iP_jP_iP_j = P_iP_iP_jP_j = P_iP_j$. This tells us that if we think of the operators P_k as being approximation schemes, then the

*This work was supported in part by the U. S. Office of Naval Research and the Air Force Office of Scientific Research under contract #N00014-80-C0176 to Drexel University.

product of two such operators provides a third and, in general, different scheme. However, as we shall see below, the derived scheme is poorer than either of its parent methods. Its approximation (interpolation) properties are, in fact, only those common to P_i and P_j. It would be nice to be able to compound P_i and P_j in such a way as to obtain an operator whose properties combined those of the two parent operators. To this end, one might consider the sum of the two operators. However, it turns out that $(P_i + P_j)(P_i + P_j) \neq P_i + P_j$; i.e., the sum of two projectors is not a projector. This essentially means that the straightforward addition of two approximation schemes does not produce another.

The combination of P_i and P_j which does combine the properties of both is the *Boolean sum* which is defined as follows: $P_i \oplus P_j = P_i + P_j - P_i P_j$. It may be easily verified that $P_i \oplus P_j$ is both linear and idempotent, and thus a projector. Since both the product and the Boolean sum of any two commutative projectors is itself a projector, the initial collection of m projectors P_i provides us a wealth of derivatives obtained by combination. Each such projector formed by combination of the P_i represents a new scheme for the approximation of functions $F \in \Psi$.

Clearly, some approximation schemes are better than others, and we can quantify this by introducing an algebraic ordering of projectors. The natural and consistent ordering relation is defined as follows: If A and B are any two projectors, then $A \leq B$ means that $AB=A$. This definition basically means that the projector (approximation scheme) B incorporates at least as much information from the function F as A does. For example, quadratic interpolation at the endpoints and midpoint of an interval is an "algebraically larger" scheme than linear interpolation to the endpoints.

When we consider the totality of all projectors which can be formed as combinations of the fundamental set $P_1, P_2, \ldots, P_m$ using the binary operations of Boolean addition and operator multiplication, we find that this collection forms a *distributive lattice* (ref. 1). Whereas every element in this lattice provides a unique approximation scheme, there are two projectors which are distinguished as being the algebraically best and worst. The best is the largest element (as measured by the above ordering relation), and is given by $P_1 \oplus P_2 \oplus \ldots \oplus P_m$. The algebraically smallest element is the product operator: $P_1 P_2 \ldots P_m$. The algebraically maximal element has the combined properties of all m of its constituents whereas the minimal element has only those properties common to all of the P_i.

There are several reasons for considering bivariate and multivariate interpolation and approximation in the somewhat abstract setting described above. One is that this consideration provides great insight into how elementary projectors can be combined to obtain very complex interpolation/approximation schemes. Secondly, it reduces the actual construction of such practical schemes to exercises in operator algebra and simple algebraic manipulation.

Yet another reason is that there exists an isomorphism relationship between the distributive lattice of projectors and the (induced) distributive lattice of *precision sets*. The precision set of the projector P_i is the set L of all linear operators L such that L P_i[F] = L[F]. For our purposes, we may think of the operators L as representing evaluation of F or some partial derivative of F at a point, along some line, or on some surface. For instance, the projector P considered below has as its precision set the two operators $L_0[F] = F(0,y)$ and

$L_1[F] = F(1,y)$. The basic result which ties together the projectors and their precision sets is the following: the precision set of a projector A in a distributive lattice is obtained from the Boolean expression for A by the following interchanges: $P_i \to L_i$; $\cdot$ (operator multiplication) $\to \cap$; $\oplus \to \cup$. Accordingly, the projector $A=(P_1 \oplus P_2)P_3$ has precision set $(L_1 \cup L_2) \cap L_3$, and $A=P_1P_2P_3$ has precision set $L_1 \cap L_2 \cap L_3$, which is clearly the smallest set that can be formed by combination of the L_i. This last observation is a reflection of the fact that $P_1P_2P_3$ is an algebraically minimal projector.

II. Sculptured Surface Interpolation Through Curve Networks

The class of problems which we consider here is of the following sort: Given a network of intersecting curves in Euclidean 3-space, construct a surface which interpolates this network. For the most part we will assume that the network is topologically rectangular.

There are two approaches to this problem: *local* and *global*. The first of these views each of the subrectangles defined by the network as a separate surface "patch" which is only weakly coupled to its neighbors. Such patch methods were first developed by S. A. Coons (ref. 2). To develop the Coons techniques it is simplest to think of the domain of the dependent variables (parameters) as being the unit square $[0,1] \times [0,1]$. We consider the two projectors: $P_s[F] = (1-s)\, F(0,t) + s\, F(1,t)$ and $P_t[F] = (1-t)\, F(s,0) + t\, F(s,1)$. It is clear that the precision set of P_s is the point set $L_s=\{s=0 \text{ and } s=1 \text{ for all } t\}$ and, similarly, the function $P_t[F]$ interpolates F along the two lines t=0 and t=1. (In practice, the function F is a vector-valued function whose components are the x, y and z coordinates of the respective curves in E^3.) It is easy to verify that these two operators are projectors and that they do commute: $P_sP_t[F] = P_tP_s[F] = (1-s)(1-t)F(0,0) + (1-s)\, t\, F(0,1) + s(1-t)\, F(1,0) + st\, F(1,1)$. Either by direct verification or by invoking the results concerning precision sets of products of operators we see that this last function interpolates F only at the four corners of the unit square. It is the familiar 4-parameter bilinear interpolant.

By forming the Boolean sum $(P_s \oplus P_t)[F] = P_s[F] + P_t[F] - P_sP_t[F]$, we obtain a *"bilinearly blended"* function which interpolates F all along the perimeter of $[0,1] \times [0,1]$. This function is the simplest of the class of *Coons patches*. In order to draw attention to the fact that the precision sets of P_s, P_t and $P_s \oplus P_t$ contain a nondenumerable number of point samples of F, they are often referred to as *transfinite interpolation schemes*.

The higher degree Coons patches are based upon Boolean combinations of odd-degree Hermite interpolation operators. For instance, if P_s is taken to be the operator, $P_s[F] = \phi_0(s)F(0,t) + \phi_1(s)\, F_s(0,t) + \phi_2(s)\, F(1,t) + \phi_3(s)\, F_s(1,t)$ where the $\phi_i(s)$ are the basis functions for cubic Hermite interpolation on $[0,1]$ and the subscript on F means partial derivative, and if P_t is taken to be the analog in the parameter t, then the *cubically blended Coons patch* is given by $(P_s \oplus P_t)[F]$. This function has the interpolatory properties of matching both F and its normal derivative all along the perimeter of the unit square. One should note that the practical implementation of this surfacing technique requires that one specify normal derivative information *at all points of the curve network* and specify so-called "twists", F_{st}, at all corners. Since such information is seldom available in design applications, this and other higher order Coons patches are seldom used. In the sequel, we shall describe global surface interpolation schemes which overcome these difficulties.

The *Coons techniques are transfinite methods* in the sense described above. In the literature, there is confusion about this, and one often finds the *16-parameter bicubic Hermite* formed by the product P_sP_t being referred to as a bicubic Coons patch. *The Coons schemes are much more general than such tensor product Hermite methods.* In the algebraic setting we see that $P_s \oplus P_t$ is greater than P_sP_t since $(P_s \oplus P_t)P_sP_t = P_sP_t$.

We return now to the original problem of interpolating a surface through a curve network. Instead of the localized, patch approach one may adopt a global approach wherein the entire network is treated as a single entity. One such method is based upon a bivariate extension of Lagrange polynomial interpolation. The relevant projectors are: $P_s[F]=\sum_{i=1}^{M}\phi_i(s)\ F(s_i,t)$ and an analogous expression for P_t. The functions $\phi_i(s)$ are the basis functions for Lagrange interpolation and M is the number of curves in the network which are parametrized by t. Since $P_s[F]$ interpolates F along each line $s=s_i$ and $P_t[F]$ does similarly along lines $t=t_i$, it should be apparent from the results of the first section that $P_s \oplus P_t$ is a transfinite projector which interpolates F along both sets of lines s_i and t_i. In other words, the function $(P_s \oplus P_t)[F]$ provides the complete solution to the network interpolation problem. For networks in which the number of curves in each direction is small (e.g., ≤ 5), this is often a viable practical solution. However, the undulating properties of univariate polynomial interpolation are inherited by this bivariate extension.

Another global scheme developed by the author and used throughout General Motors' CAD/CAM systems is the technique of *cubic spline-blended interpolation.* The projectors associated with this method are essentially the same as the above Lagrange projectors except that the blending functions are cubic splines in place of polynomials. The surfaces generated by this technique inherit the benign properties of univariate cubic splines (e.g., smoothness and C^2 continuity) and have been found to be fully satisfactory by designers and manufacturing engineers.

For a more detailed presentation of the results discussed in this section, see reference 3.

III. Solid Modeling

Space does not permit a detailed account of how one uses the results of Sec. I to model 3-D solid structures. This is unfortunate since the distributive lattice associated with three commutative projectors, P_s, P_t and P_u, is much richer and mathematically more interesting than the simple 4-element lattice generated by two projectors, as in the previous section.

The lattice theoretic techniques described above can be used to derive expressions for the modeling of any simply connected solid which is parametrized so as to have six faces, some of which may be degenerate (i.e., a curve or a point). If one takes the projector P_s to be $P_s[F]=(1-s)F(0,t,u)+s\ F(1,t,u)$ and takes P_t and P_u to be the analogous linear interpolation operators in t and u, then it is easy to show that the algebraically maximal projector in the lattice, $P_s \oplus P_t \oplus P_u$, corresponds to interpolation to F on all of the six faces of the cube $[0,1]\times[0,1]\times[0,1]$. The minimal projector $P_sP_tP_u$ interpolates only at the eight corners. Other combinations of the three generators interpolate at various combinations of faces, edges and corners. As in the bivariate case, the choice of an appropriate projector depends upon the data structure (i.e., what geometric information is given) and such considerations as required smoothness and ultimate use of the model.

References

1. W.J. Gordon, "Distributive Lattices and the Approximation of Multivariate Functions," in Approximations with Special Emphasis on Spline Functions, Academic Press, New York, 1969, pp. 223-277.

2. S.A. Coons, "Surfaces for Computer-Aided Design of Space Forms," MAC-TR-41, Massachusetts Institute of Technology, 1967. (Available from DDC as AD663504).

3. W.J. Gordon, "Sculptured Surface Interpolation via Blending-Function Methods," Drexel University Research Report, 1982.

A GEOMETRIC MODELER BASED ON A DUAL-GEOMETRY REPRESENTATION - POLYHEDRA AND RATIONAL B-SPLINES

Albert L. Klosterman
Structural Dynamics Research Corporation
Milford, Ohio

In reviewing the current literature one notices that for speed and data base reasons, solid geometric modeling of large complex practical systems is usually approximated by a polyhedra representation. Precise parametric surface and implicit algebraic modelers are available but it is not yet practical to model the same level of system complexity with these precise modelers. Compare, for example, the typical complex system shown in Figure 1 for a polyhedra modeler versus the single components found in the literature (ref. 1) (Fig. 2) for algebraic or parametric modelers. This contrast led SDCR to believe that a new modeler which encompassed both of these geometric representations would be appropriate. Therefore the GEOMOD Geometric Modeling System was built so that a polyhedra abstraction of the geometry is available for interactive modeling without losing the precise definition of the geometry (ref. 2).

Part of the reason that polyhedra modelers are effective in modeling complex systems is that all bounded surfaces can be represented in a single canonical format (i.e. sets of planar polygons). This permits a very simple and compact data structure. Another reason is due to the relatively easy implementation of a wide variety of highly interactive modeling techniques like:

1. Large number of primitives
2. Extrusion operations
3. Revolution operations
4. Skinning operations
5. Boolean operations
6. Deformation operations (i.e. bending, warping, blending, etc.)

Therefore, in building GEOMOD it was important to accommodate these strategic elements.

Nonuniform rational B-splines (refs. 3-5) are currently the best representation to describe a very large class of geometry precisely with one canonical format. For example, they have the following advantages:

- One basic canonical form is used to represent all the curve and surface entities with a straightforward data structure.

- Transformations, such as translations, rotations, mirror images or scaling can be represented by preforming the corresponding transformations on the control points of the original entity.

- This form is well suited for fast and accurate parametric evaluation to find points on the entity and to obtain derivatives of all orders.

- All surfaces represented as rational B-splines can use the same routines for intersection, orientation, display, area, volume, normals, tangents, etc.

The objective of allowing a wide variety of data-base-efficient and highly interactive modeling and display techniques is difficult to accomplish if the modeler deals only with precise surfaces. However, this objective can be met if the user has a polyhedra model available to work with interactively and to display quickly, and which defines the topology of the object necessary for use of the precise surfaces when required. This relieves the requirement to calculate and store a complete, precise, evaluated boundary file model at each modeling step. Also, a number of the difficult deformation operators are easily accomplished if a general surface fitting routine is provided which allows the user to deform the polyhedra model and then fit the resulting surfaces with rational B-splines.

With this approach to building a modeler, all the advantages of polyhedra-based systems are obtained. The following capabilities can also make use of the precise surface geometry:

- Automatic refinement of the approximate surfaces at any time for a more accurate polyhedra representation

- Display of smooth, precise line drawings when necessary

- Calculation of precise area, mass, and inertia properties

- Creation of a precise geometric boundary file model for use by detailed design and manufacturing functions

For example, Figure 3 shows the polyhedra result of subtracting three orthogonal cylinders from a block. This representation is quite effective for user interaction and is frequently appropriate for description of this component in an overall system. However, the precise description shown in Figure 4 must be callable on demand even though it takes longer to compute. Similarly, Figure 5 shows a typical 2-D profile of an airfoil defined by specifying standard 2-D curve entities (i.e. lines, arcs, splines, etc.). These standard entities are represented as nonuniform rational B-spline curves before they are extruded, revolved, or skinned to create a 3-D object. Figure 6 shows a wire frame representation of the non-uniform rational B-spline surfaces for the airfoil created by skinning a set of cross sections. Figure 7 is the corresponding polyhedra representation created automatically with a user-specified level of accuracy. This polyhedra abstraction of the actual object can be utilized for further interactive user interface and can also provide a quick approximate method to determine the visibiliy of the precise surface intersection lines shown in Figure 8.

The experience to date indicates that this dual-geometry representation has allowed a system to be developed which permits both modeling of complex systems and precise definition of components to coexist in the same interactive environment.

References

1. Requicha, A. G.: Representations for Rigid Solids: Theory, Methods, and Systems. Computing Surveys, vol. 12, no. 4, Dec. 1980.

2. Klosterman, A. L.; Ard, R. H.; and Klahs, J. W.: A Geometric Modeling Program for the System Designer. Presented at Conference on CAD/CAM Technology in Mechanical Engineering, Mass. Institute of Technology, 1982.

3. Versprille, K. J.: Computer-Aided Design Applications of the Rational B-Spline Approximate Form. Ph.D Dissertation, Syracuse University, 1975.

4. Blomgren, R. M., and Fuhr, R. D.: Rational B-Spline Curves. Initial Graphics Exchange Specifications (IGES) Version 2.0, Appendix A4, NBSIR-82-2631(AF), National Bureau of Standards, 1982.

5. Blomgren, R. M.; and Fuhr, R. D.: Rational B-Spline Surfaces. Initial Graphics Exchange Specifications (IGES) Version 2.0, Appendix A6, NBSIR-82-2631(AF), National Bureau of Standards, 1982.

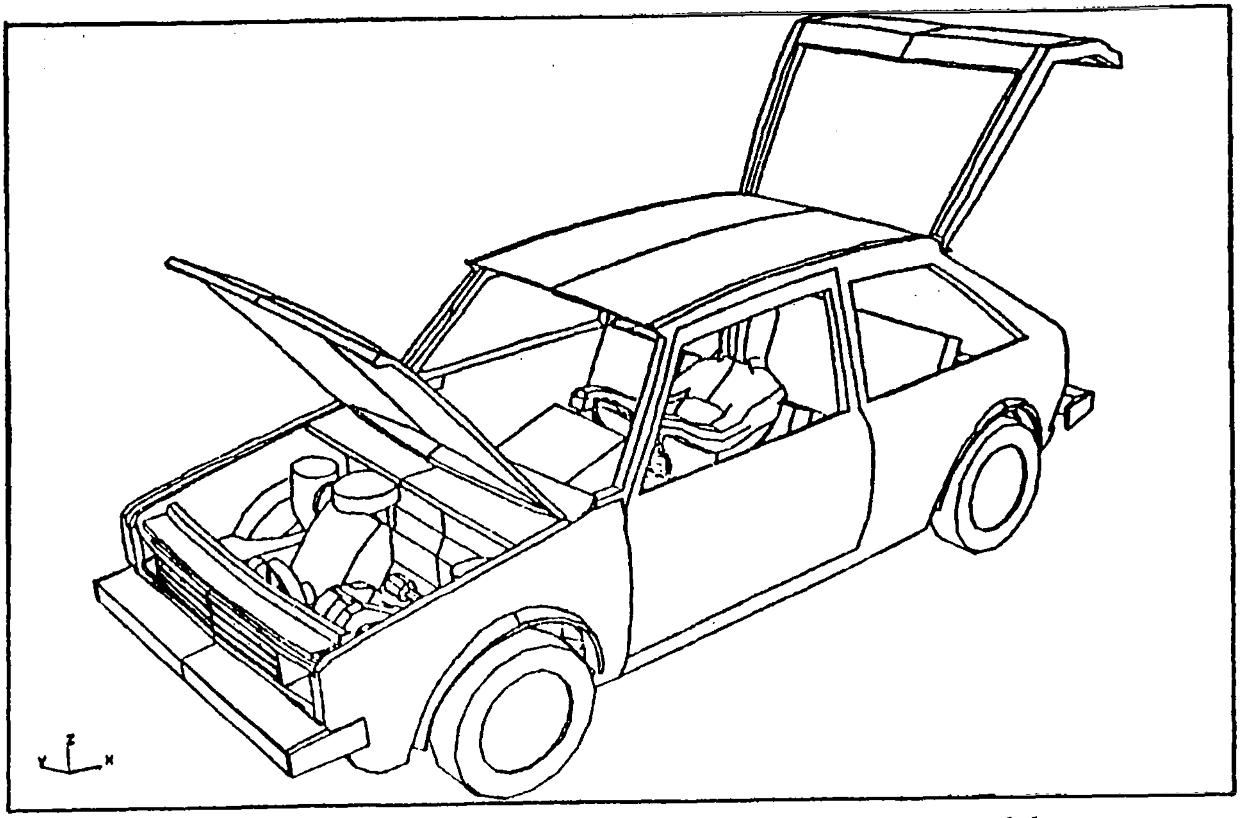


Figure 1. - Typical complex system modeled with a polyhedra modeler.

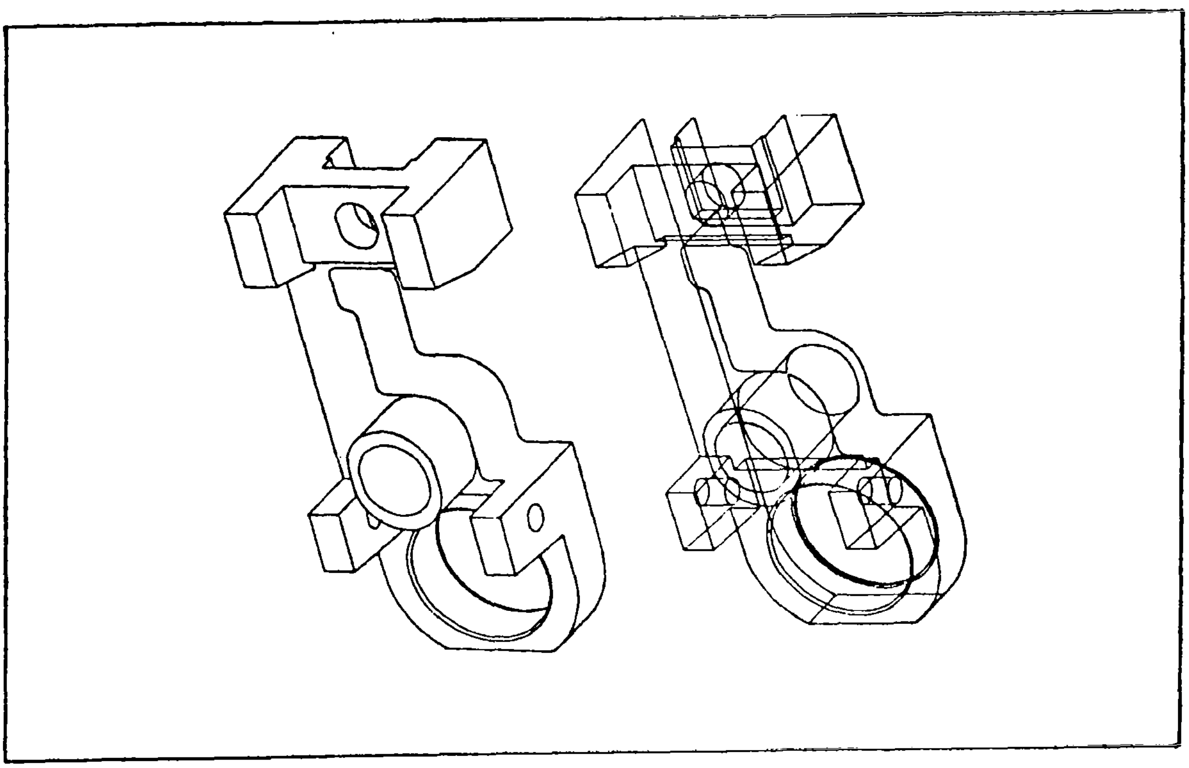


Figure 2. - Typical component used to demonstrate current precise modelers.

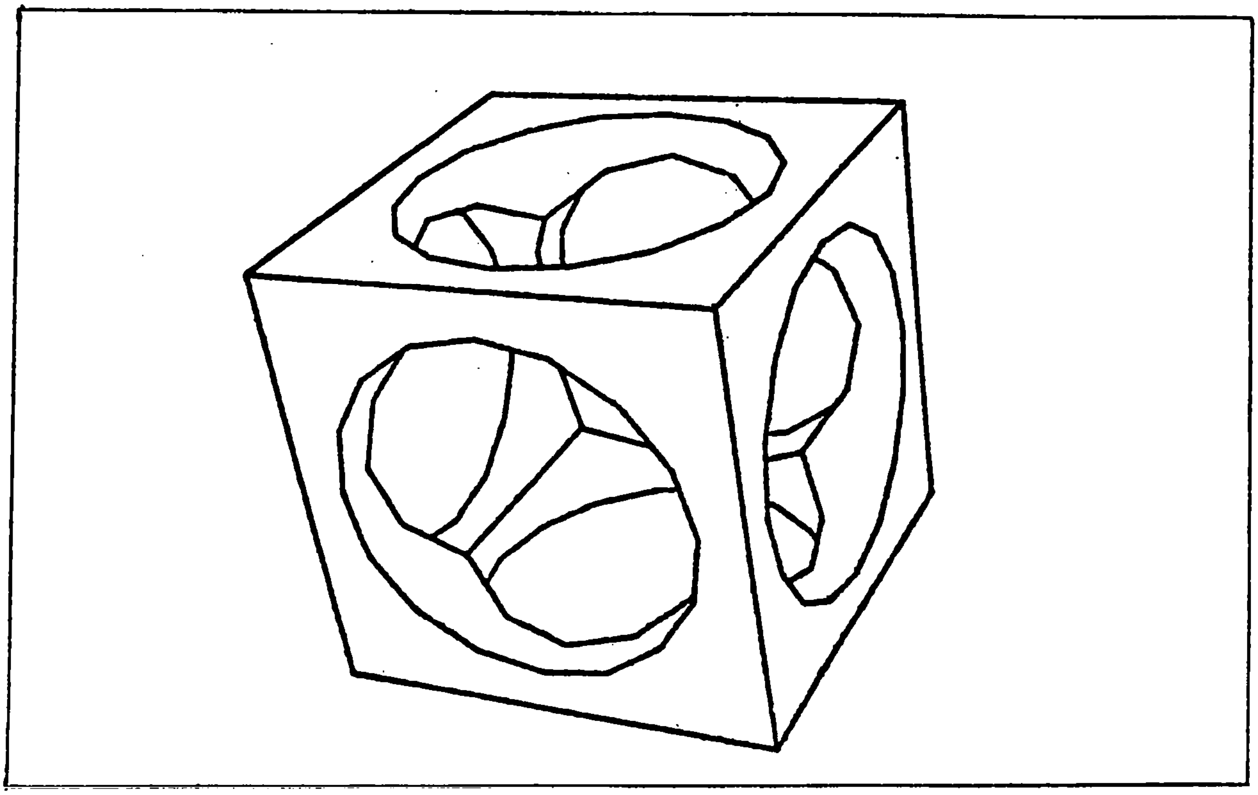


Figure 3. - Polyhedra representation of block cut by 3 cylinders.

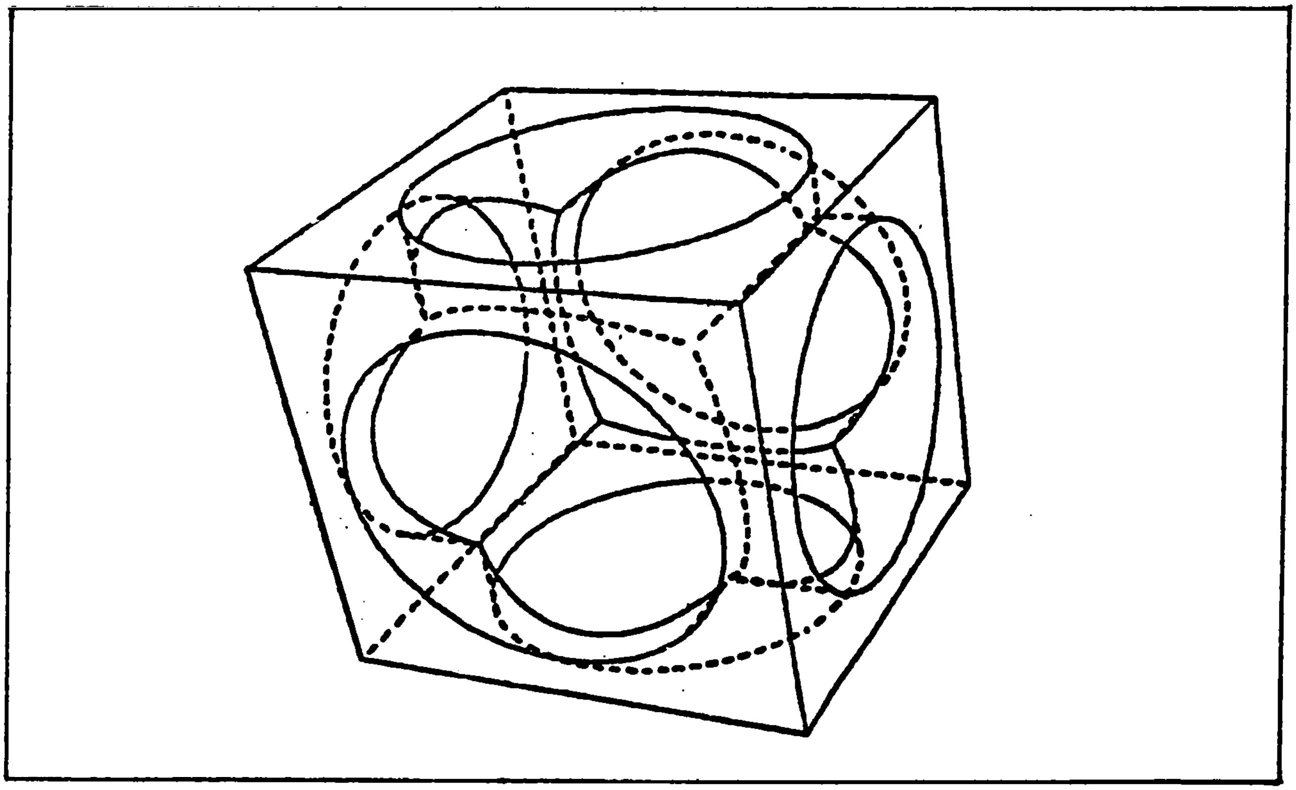


Figure 4. - Rational B-spline representation of block cut by 3 cylinders.

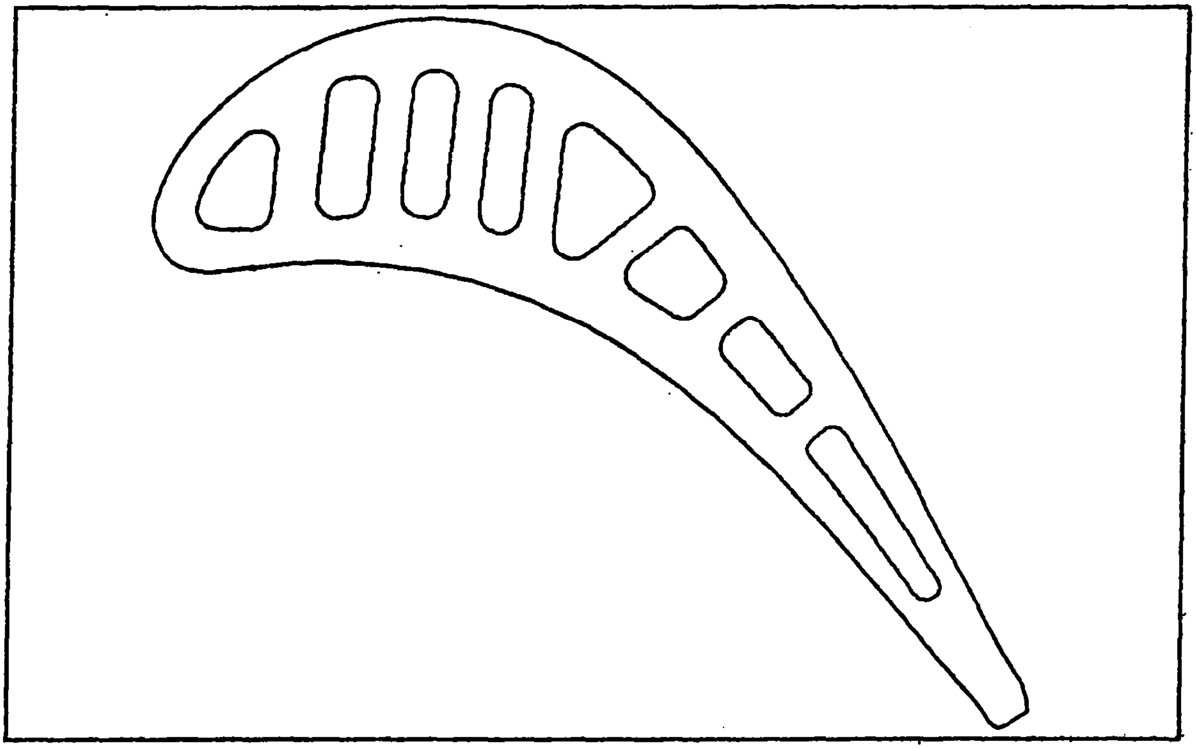


Figure 5. - Two-dimensional rational B-spline representation of a cross section of airfoil.

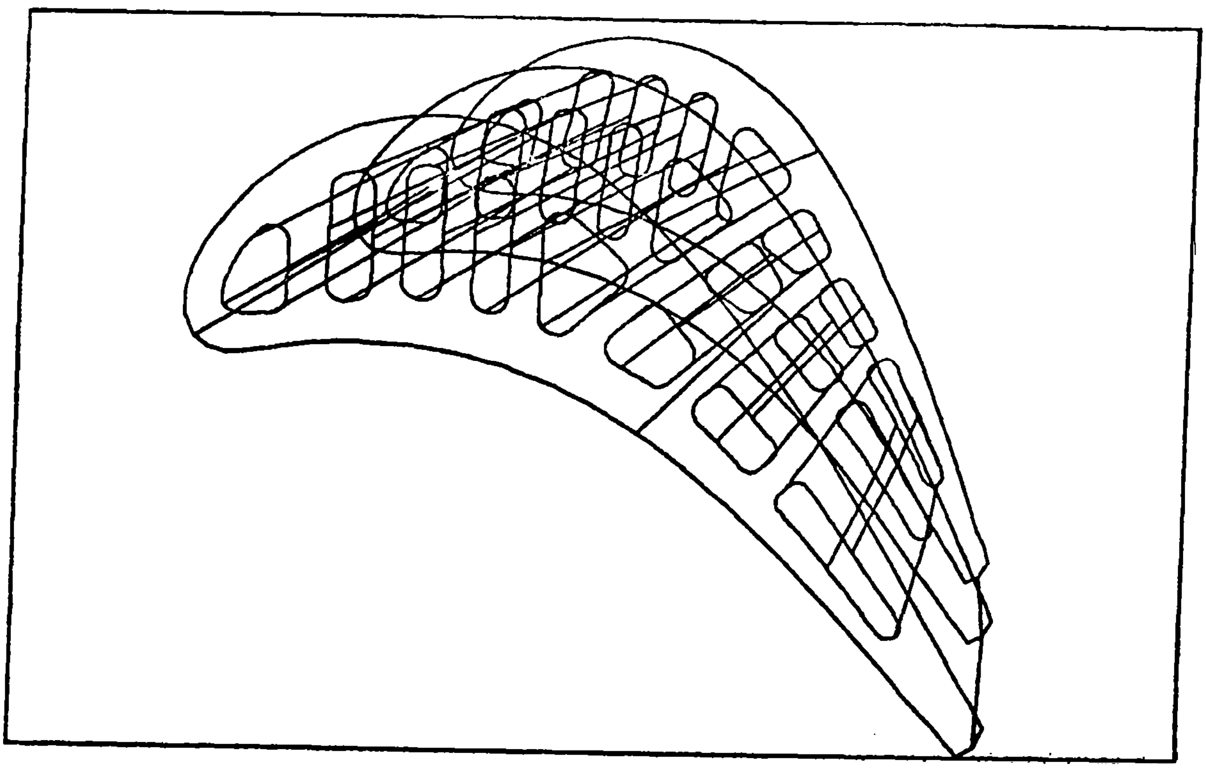


Figure 6. - Wire frame presentation of rational B-spline representation of 3-D airfoil.

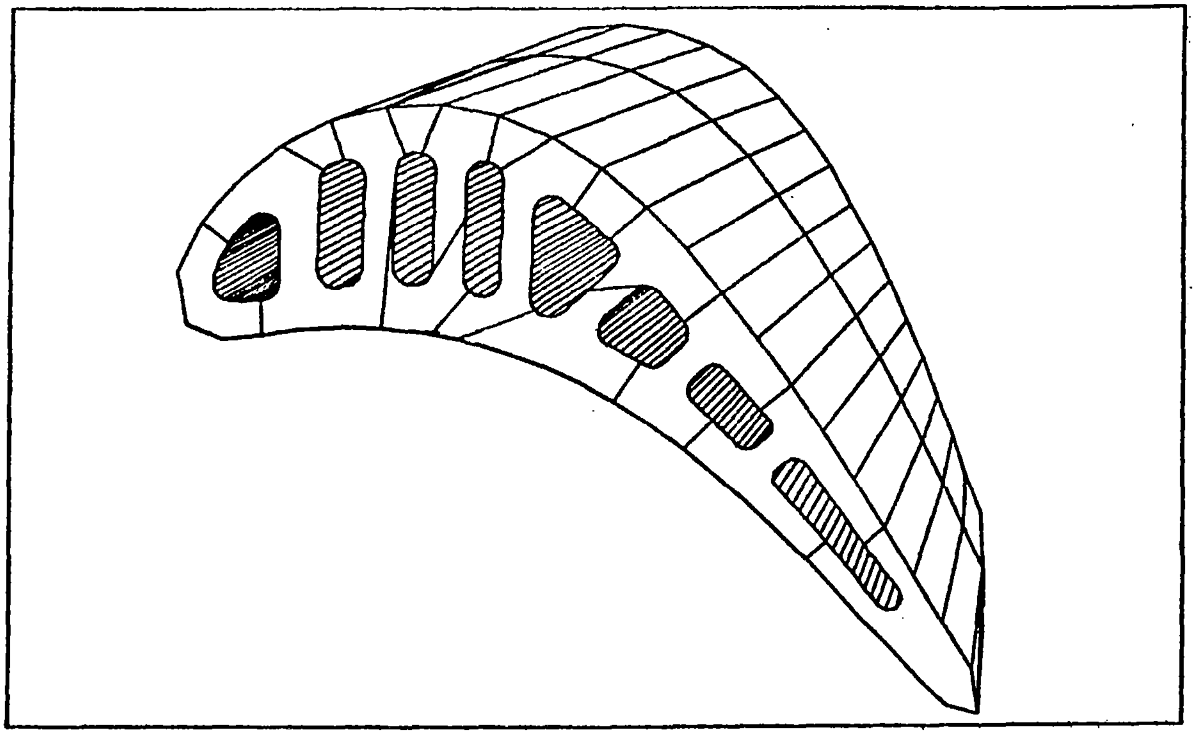


Figure 7. - Hidden-line presentation of the polyhedra representation of airfoil.

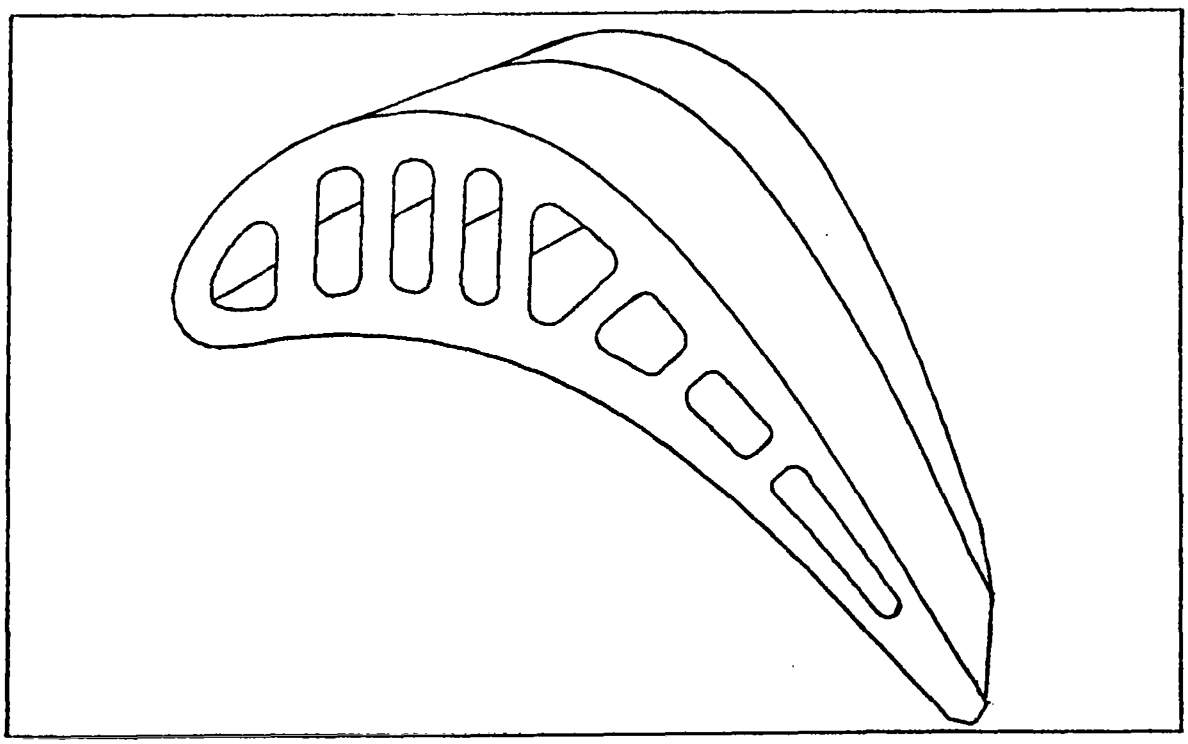


Figure 8. - Hidden-line presentation of the rational B-spline representation of airfoil.

THE COMPUTATION OF ALL PLANE/SURFACE INTERSECTIONS FOR CAD/CAM APPLICATIONS

David H. Hoitsma, Jr.
Fairchild Republic
Farmingdale, New York

Martin Roche
Lundy Electronics & Systems Inc.
Glen Head, New York

This paper focuses on the problem of the computation and display of all intersections of a given plane with a rational bicubic surface patch for use on an interactive CAD/CAM system. The general problem of calculating all intersections of a plane and a surface consisting of rational bicubic patches is reduced to the case of a single generic patch by applying a rejection algorithm which excludes all patches that do not intersect the plane.

For each pertinent patch the algorithm presented computes the intersection curves by locating an initial point on each curve, and computes successive points on the curve using a tolerance step equation similar to that given in reference 1. A single cubic equation solver is used to compute the initial curve points lying on the boundary of a surface patch, and the method of resultants as applied to curve theory (ref. 2) is used to determine critical points which, in turn, are used to locate initial points that lie on intersection curves which are in the interior of the patch.

Examples are given to illustrate the ability of this algorithm to produce all intersection curves. One example (Figure 1) show how more than one curve segment can be determined by a patch-plane intersection and have no critical points. Another example (Figure 2) illustrates a patch being cut by a plane with a critical point relative to the plane; the patch boundary curve segments and the plane do not intersect.

The plane/surface intersection problem is stated and a procedure is outlined that will determine initial points on all intersection curve segments for a given Coons patch. In particular the rational bicubic Coons' surface P (u,v) is a collection of patches such that

$$P(u,v) = P_{rs}(u,v), \quad r = 1,\ldots,R, \quad s = 1,\ldots,S$$

where

$$P_{rs}(u,v) = (X_{rs}(u,v), Y_{rs}(u,v), Z_{rs}(u,v), W_{rs}(u,v)) = \sum_j \sum_i A_{ij}(r,s)\, u^i v^j$$

with $0 \leq u \leq 1$, $0 \leq v \leq 1$ and each $A_{ij}(r,s)$ is a 4-dimensional vector.

A rejection algorithm based on patch bounds (ref. 3) or recursive subdivision methods (ref. 4) will exclude patches which do not intersect the plane and reduces the problem to the calculation of the intersection curves of the remaining patches.

Let U denote the unit square $0 \le u \le 1$, $0 \le v \le 1$, and without loss of generality let the cutting plane by $Z = 0$. A typical surface patch is described by

$$P\ (u,v) = (X\ (u,v),\ Y\ (u,v),\ Z\ (u,v),\ W\ (u,v)) = \sum_{j=o}^{3} \sum_{i=o}^{3} A_{ij}\ u^i\ v^j$$

with $A_{ij} = (X_{ij}, Y_{ij}, Z_{ij}, W_{ij})$.

The problem is to find all solution curves of

$$\text{(1)} \quad Z\ (u,v) = \sum_{j=o}^{3} \sum_{i=o}^{3} Z_{ij} u^i v^i = 0 \qquad 0 \le u \le 1, \quad 0 \le v \le 1 \tag{1}$$

There are two types of solution curves for equation (1). Either the curve starts and ends on the boundary of U or it is a closed curve in the interior of U. In considering these two types of curves the following steps are taken.

Step 1. Locate the critical points of $Z\ (u,v) = 0$ by solving the system

$$p\ (u,v) = \frac{\partial Z}{\partial u} = 0$$

$$q\ (u,v) = \frac{\partial Z}{\partial v} = 0$$

Consider the resultant $R(u)$ of p and q with respect to v. The zeros of the resultant $R(u)$ give those values u_i, $i = 1,\dots,m$ such that the common roots v_{ij}, $j = 1,\dots,n_i$, of $p\ (u_i,v) = 0$ and $q\ (u_i,v) = 0$ completely determine the critical points. That is, the pairs (u_i,v_{ij}) so determined are the critical points of $Z\ (u,v) = 0$. The values u are found by using a polynomial root finder and the values v_{ij} are determined by the Euclidean algorithm for the greatest common divisor.

Step 2. Determine which of the critical points are local maxima and minima of $Z\ (u,v)$.

Step 3. Use the local extrema values to determine points on closed intersection curves (and compute closed-intersection curves). If (u^*, v^*) is a local extremum, the roots of the cubic equation $Z\ (u,v^*) = 0$ will determine possible closed-intersection curve points. This determination is true since each closed-intersection curve contains a local maximum or minimum (Ref. 5).

Step 4. Use the plane/patch boundary intersection points as initial points (and compute the intersection curve). Note that not all of these intersection points will be used as initial points since most curve segments starting at a boundary point will finish at a different boundary point.

Using this procedure a point on all curve segments will be found and hence all curve segments will be computed.

In addition to the above-mentioned references, other pertinent references are references 6 through 9.

REFERENCES

1. Faux, I. D.; and Pratt, M. J.: Computational Geometry for Design and Manufacture. Chapter 9, John Wiley & Sons, 1979.

2. Walker, R.: Algebraic Curves. Dover Publications, Inc., New York, 1950.

3. Dimsdale, B.; and Burkley, R. M.: Bicubic Patch Surfaces for High-Speed Numerical Control Processing. IBM J. Res. and Devel., 1976, pp. 358-367.

4. Lane, J. M.; and Riesenfeld, R.: A Theoretical Development for the Computer Generation and Display of Piecewise Polynomial Surfaces. IEEE Trans. Pattern Analysis and Machine Intelligence, vol. PAMI-2, no. 1, Jan. 1980, pp. 35-46.

5. Buck, R. C.: Advanced Calculus. McGraw-Hill Book Company, Inc., New York, 1956.

6. Hyodo, Yoshihiro: HAPT-3D: A Programming System for Numerical Control. Computer Languages for Numerical Control, Proc. 2nd IFIP/IFAC International Conf. on Programming Languages for Machine Tools, J. Hatvany, ed., North Holland Publ. Co., Amsterdam, 439-448.

7. Forrest, A. R.: Curves and Surfaces for Computer-Aided Design. Ph. D. Thesis, Chapter 13, University of Cambridge, UK, 1968.

8. Lee, Theodore M. P: Three-Dimensional Curves and Surfaces for Rapid Computer Display. ESD-TR-69-189, Section 3, Air Force Systems Command, 1969.

9. Kajiya, J. T.: Ray Tracing Parametric Patches. Computer Graphics, vol. 16, 1982.

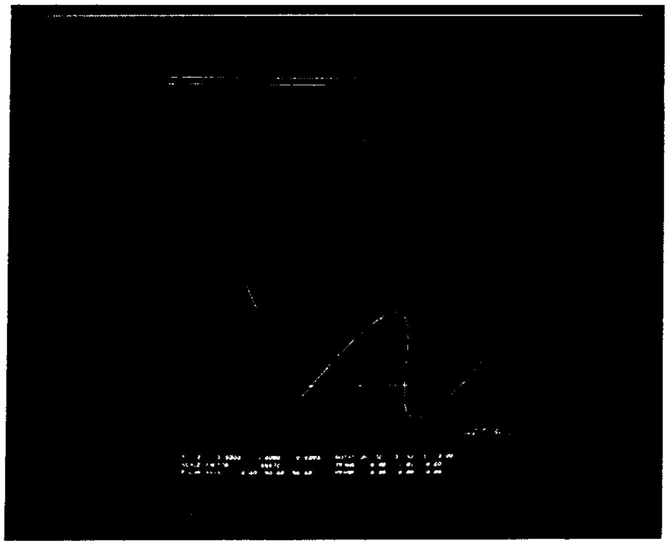


Figure 1.- Plane-patch intersection with
no critical points.

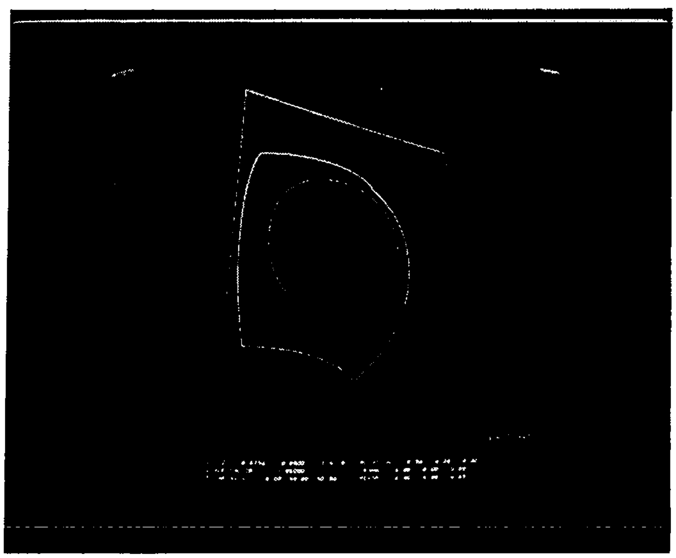


Figure 2.- Plane-patch intersection with
a critical point.

USE OF SHAPE-PRESERVING INTERPOLATION METHODS IN SURFACE MODELING

F. N. Fritsch*
Lawrence Livermore National Laboratory
Livermore, California

In many large-scale scientific computations, it is necessary to use surface models based on information provided at only a finite number of points (rather than determined everywhere via an analytic formula). As an example, an equation of state (EOS) table may provide values of pressure (P) as a function of temperature (T) and density (d) for a particular material. These values, while known quite accurately, are typically known only on a rectangular (but generally quite nonuniform) mesh in (T,d)-space. Thus interpolation methods are necessary to completely determine the EOS surface.

The most primitive EOS interpolation scheme is bilinear interpolation. This has the advantages of depending only on local information, so that changes in data remote from a mesh element have no effect on the surface over the element, and of preserving shape information, such as monotonicity. Most scientific calculations, however, require greater smoothness; i.e., at least first derivative continuity. Standard higher-order interpolation schemes, such as Coons patches or bicubic splines, while providing the requisite smoothness, tend to produce surfaces that are not "physically reasonable". This means that the interpolant may have "bumps" or "wiggles" that are not supported by the data. This is particularly true of EOS surfaces for materials which experience phase transitions in the region of interest, so that the surface is constant as a function of one of the independent variables in a certain subregion.

Let us first consider some one-dimensional examples. Given a set of points which are to define a design curve, a common practice is to pass a piecewise cubic curve through the points that is "fair" or "visually pleasing". If one defines "fair" to mean continuous curvature and minimum "energy", the result is a cubic spline. As we see in Figure 1, an interpolating cubic spline need not be "physically reasonable". In this case, the quantity being modeled must have values between zero and one to be physically meaningful. The cubic spline clearly does not satisfy these conditions.

---

* Presently at Drexel University, Philadelphia, Pennsylvania.

In order to meet these objections, it was decided to consider a more general piecewise cubic Hermite (PCH) interpolant. Such a curve is a cubic polynomial between the data points, with the pieces joined so that the function and its first derivative (tangent) are continuous. A PCH function is determined uniquely by its values and the values of its slope at the breakpoints. The cubic spline is obtained if we require that the function also have continuous second derivative (curvature) at the breakpoints. In retrospect, we see that failure to preserve the increasing nature of the data of Figure 1 has led to the oscillations that caused violation of the physical constraints. If we are willing to give up second derivative continuity, Fritsch and Carlson [1] have shown that it is possible to determine suitable derivative approximations to preserve monotonicity of the data. Figure 2 shows the result of applying their method, as modified for better locality by Fritsch and Butland [2], to these same data. It would appear that preserving monotonicity is more "physically reasonable" than second derivative continuity in this case.

In Figures 3-6 the data are only piecewise monotonic. The independent variable (time) values are uniformly spaced. The physical quantity being modeled is always nonnegative. It starts out at zero and increases sharply to a maximum. It then decreases until it reaches a plateau, is constant for a while, and then drops back to zero. In Figures 3 and 4 we see how unsatisfactory a cubic spline can be. In particular, the second derivative continuity makes it impossible for the interpolant to be constant unless it is constant everywhere. The only difference between the two curves is the boundary conditions. Figure 3 is the "natural" cubic spline, obtained by setting the second derivative to zero at the boundaries. In Figure 4 we have attempted to prevent the curve from going negative by specifying a zero slope at the initial point. If we apply subroutine PCHIM from the new interpolation package PCHIP [3,4] to Pruess' data [5] we obtain Figure 5. This is much more "physically reasonable", except that requiring strict monotonicity on each interval has flattened out the peak in an unacceptable way. If we instead apply PCHIC, which provides special treatment in the vicinity of extrema in the data, we obtain Figure 6.

These same ideas can be extended to surfaces if the defining data points lie on a rectangular mesh, as is typically the case for EOS data. We shall consider piecewise bicubic Hermite (PBH) interpolants. These are determined by values of the function, its first partial derivatives, and its cross derivative (twist) at the mesh points. A PBH surface has continuous tangents across the mesh lines. If we require second derivative continuity, we obtain a bicubic spline.

In Figure 7, we show a portion of an EOS surface for aluminum, with its bicubic spline interpolant. This is clearly unacceptable! In Figure 8 we have used standard three-point formulas to approximate the first partial derivatives and the twists. The resulting surface is better, but still contains unacceptable overshoots in the vicinity of the phase transition. In Figure 9 we have used the one-dimensional monotonicity preserving interpolation routine PCHIM for the first partials and set the twists to zero. This surface is monotonic along the mesh lines, but close examination reveals that the surface still fails to be monotonic in the interior of some of the mesh boxes. Application of an experimental algorithm by Carlson and Fritsch [6] to this EOS produces the surface in Figure 10. This algorithm can handle many EOS-like surfaces, but further work is needed to improve its efficiency and enable it to treat data that are only piecewise monotonic.

My main purpose for talking at this symposium is to generate discussion that may eventually lead to mathematical quantification of the ideas of "fair", "visually pleasing", or "physically reasonable" curves and surfaces. These examples clearly demonstrate that the traditional concepts of mathematical smoothness and energy minimization may not be the right measures.

Acknowledgement. This work was supported by the U.S. Department of Energy under Contract W-7405-Eng-48 and by its Office of Basic Energy Sciences, Applied Mathematical Sciences Program.

REFERENCES:

1. Fritsch, F. N., and Carlson, R. E., "Monotone Piecewise Cubic Interpolation", SIAM J. Numer, Anal., 17, 2 (April 1980), 238-246.

2. Fritsch, F. N., and Butland, J., "A Method for Constructing Local Monotone Piecewise Cubic Interpolants", LLNL Preprint UCRL-87559, Lawrence Livermore National Laboratory (April 1982). [To appear in SIAM J. Sci. Stat. Comput.]

3. Fritsch, F. N., "PCHIP Final Specifications", LLNL Computer Documentation Report UCID-30194, Lawrence Livermore National Laboratory (August 1982).

4. Fritsch, F. N., and Carlson, R. E., "Piecewise Cubic Hermite Interpolation Package", LLNL Preprint UCRL-87285, Lawrence Livermore National Laboratory (July 1982).

5. Pruess, S., An Algorithm for Computing Smoothing Splines in Tension. Computing, 19 (1978), 365-373.

6. Carlson, R. E., and Fritsch, F. N., "Monotone Piecewise Bicubic Interpolation", LLNL Preprint UCRL-86449, Lawrence Livermore National Laboratory, 1981.

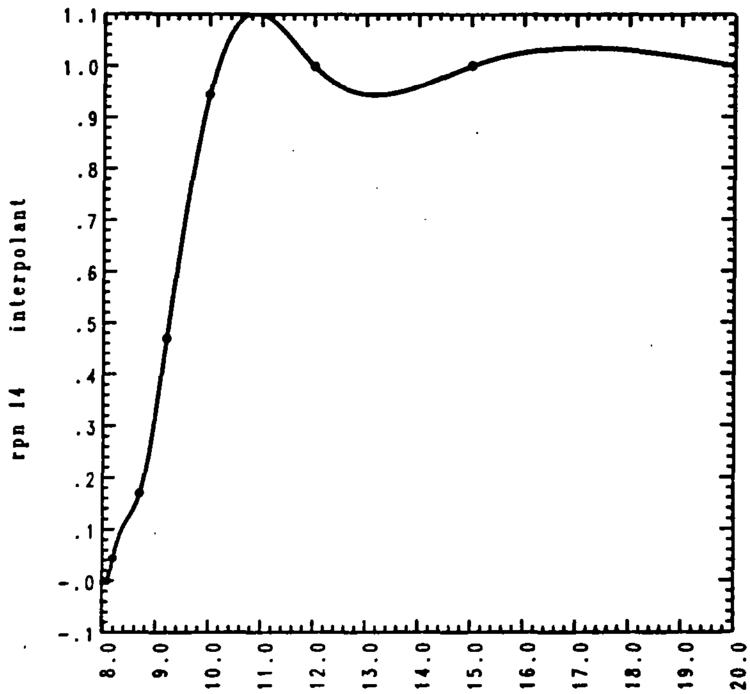


Figure 1.- Natural cubic spline interpolant to RPN 14 data of [1].

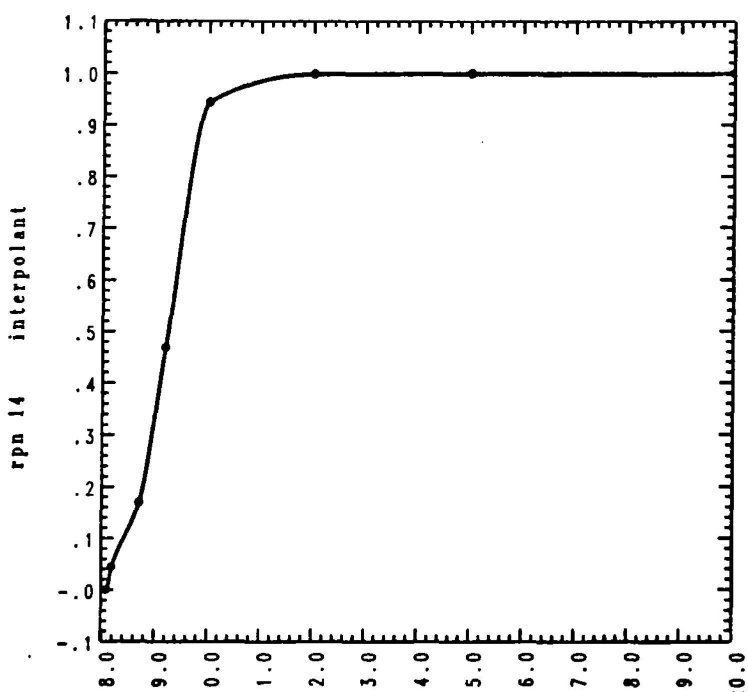


Figure 2.- Fritsch-Butland [2] interpolant to these same data.

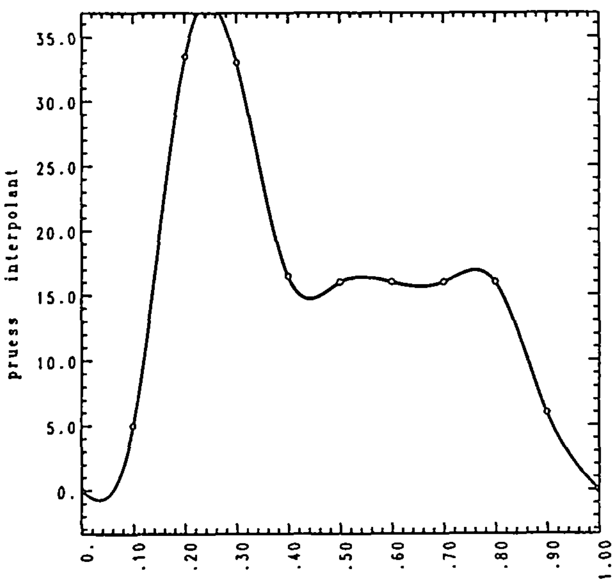


Figure 3.- Natural cubic spline interpolant to Pruess [5] data.

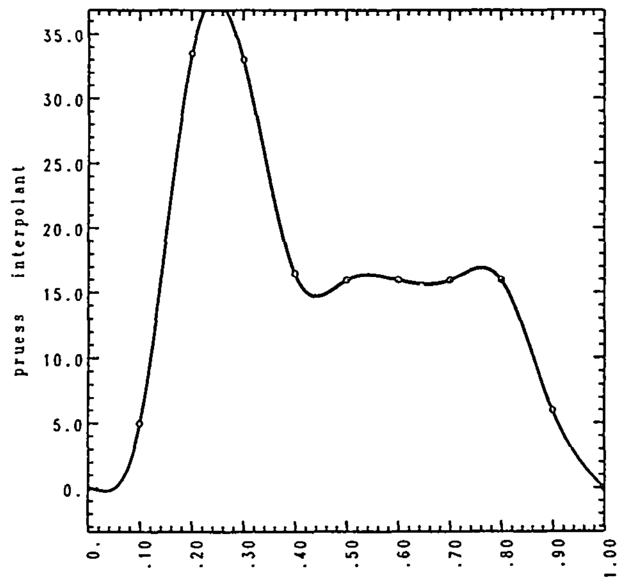


Figure 4.- Cubic spline with $s'(a) = 0$, $s'(b) = -40$.

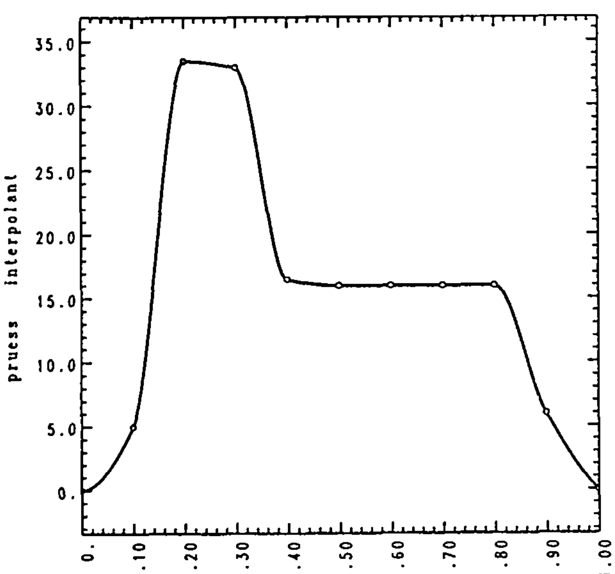


Figure 5.- PCHIM [3] interpolant to these same data.

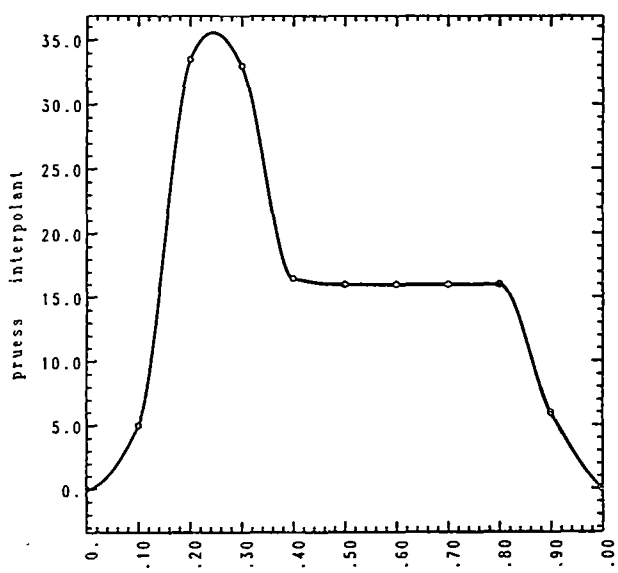


Figure 6.- PCHIC [3] interpolant (using defaults).

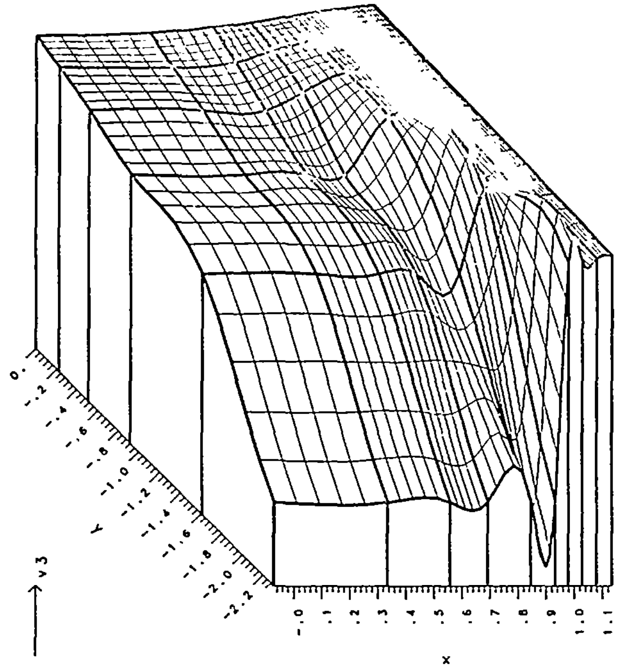


Figure 7.- Bicubic spline interpolant to an aluminum EOS.

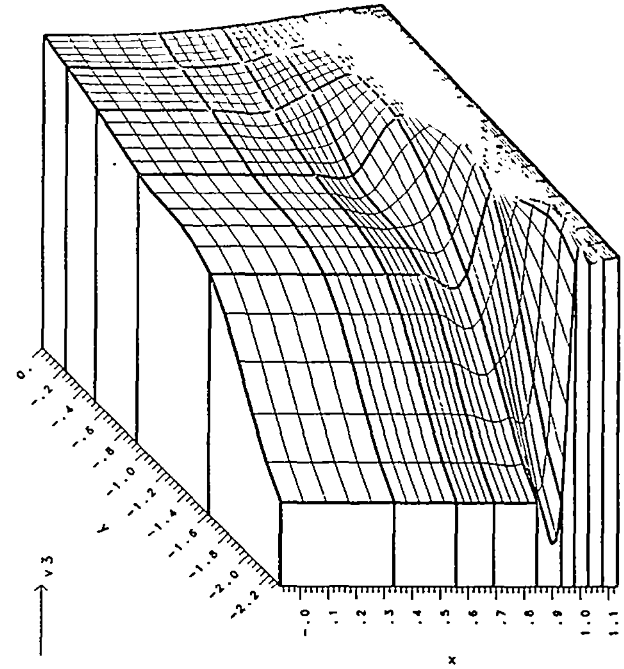


Figure 8.- Bicubic Bessel (3-point difference) interpolant.

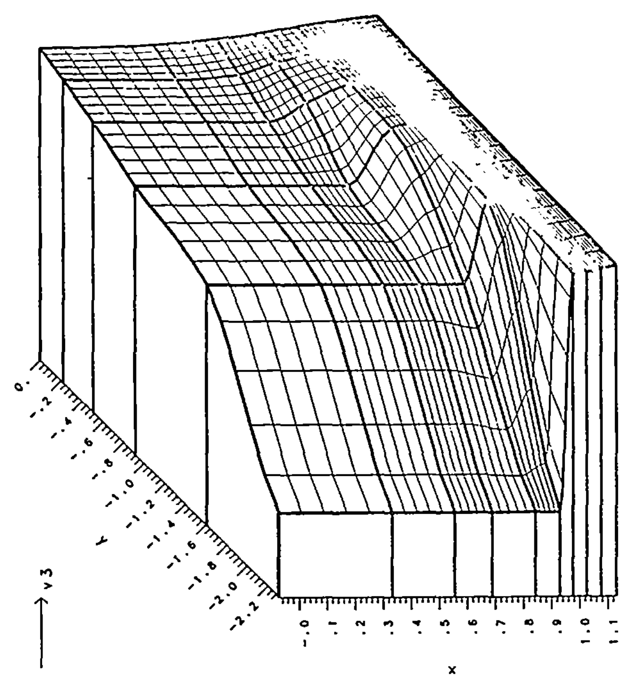


Figure 9.- PBH interpolant using PCHIM [3] for first partial derivatives, zero twists.

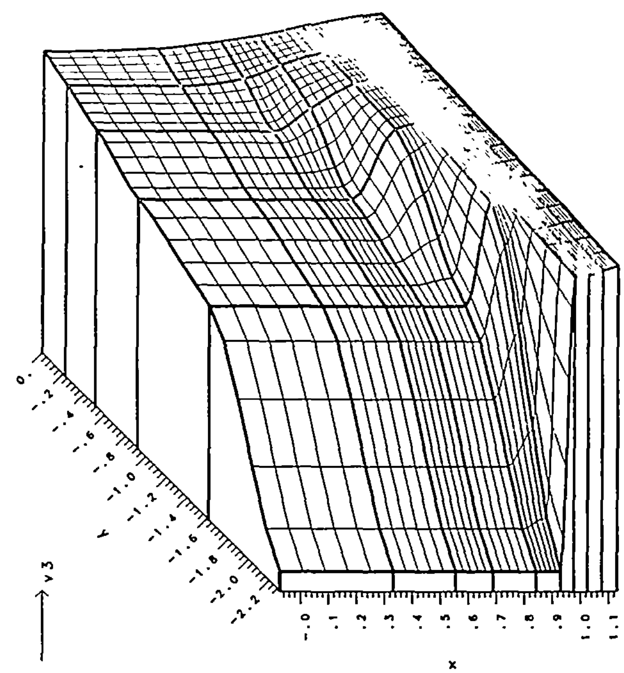


Figure 10.- Monotonicity-preserving PBH interpolant [6].

AN URNFUL OF BLENDING FUNCTIONS

R. N. Goldman
Control Data Corporation
Minneapolis, Minnesota

1. Introduction

There is a deep, fundamental connection between the mathematical theory of discrete probability distributions and the parametric curves and surfaces of computer-aided geometric design. It is no accident that the blending functions of Bezier curves and surfaces have an obvious probabilistic interpretation, nor is it a coincidence that the normalized uniform B-spline basis functions also model a simple stochastic process (see section 5). In this paper we shall explore further this link between probability and geometry, and we shall show how to exploit simple probabilistic arguments to derive many of the classical geometric properties of the parametric curves and surfaces currently in vogue in computer-aided geometric design. We shall also use this probabilistic approach to introduce many new types of curves and surfaces, and we shall demonstrate how probability theory can be used to simplify, unify, and generalize many well-known results.

2. Motivation

Typically in computer-aided geometric design a designer specifies only a relatively small collection of control points. The burden then shifts to the program to blend these points into the desired curve or surface. Thus the canonical problem in computer-aided geometric design is: given an array of control points $P = (P_j)$ or $P = (P_{jk})$, generate a curve or surface which conforms to the shape described by these points. The standard solution to this canonical problem is to endow the program with a collection of blending functions $B(t) = (B_j(t))$ or $B(s,t) = (B_{jk}(s,t))$, and to define a parametric curve or surface by setting

$$B[P](t) = \sum_j B_j(t)\, P_j \qquad 0 \le t \le 1 \text{ (curve)}$$

$$B[P](s,t) = \sum_{j,k} B_{jk}(s,t) P_{jk} \qquad 0 \le s,t \le 1 \text{ (surface)}$$

Since the curve or surface must conform to the shape defined by the original control points, the blending functions cannot be arbitrary; indeed they must have certain very special characteristics. We shall list many of these characteristic properties explicitly in section 4, but here we shall confine our attention to the two most basic properties.

In computer aided geometric design the parametric curve or surface must depend only on the designer's control points. In particular, it must be independent of the internal coordinate system of the program. Indeed, in general, the designer is unaware of this coordinate system, and he would soon become very perplexed if the same collection of control points was suddenly to generate a different curve or surface merely due to some internal change in the coordinate system. In order for the curves and surfaces to be independent of the choice of the coordinate origin, the blending functions must satisfy the identity

$$\sum_j B_j(t) = 1 \qquad \text{for all } t \qquad \text{(curves)}$$

$$\sum_{j,k} B_{jk}(s,t) = 1 \qquad \text{for all } s,t \qquad \text{(surfaces)}$$

When this condition is satisfied, we shall say that the curves and surfaces are well defined.

A designer expects not only the shape but also the location of his curve or surface to be regulated by his control points. The standard way to assure that the curve or surface lies in the general proximity of its control points is to confine it to the convex hull of these points. Now a well-defined curve or surface will lie in the convex hull of its control points if and only if

$$B_j(t) \geq 0 \qquad \text{for all } j \qquad \text{(curves)}$$

$$B_{jk}(s,t) \geq 0 \qquad \text{for all } j,k \qquad \text{(surfaces)}$$

for $0 \leq s,t \leq 1$.

The two conditions

1. $\sum_j B_j(t) = 1$ or $\sum_{j,k} B_{jk}(s,t) = 1$

2. $B_j(t) \geq 0$ or $B_{jk}(s,t) \geq 0$

are the defining characteristics of discrete probability distributions. Therefore, if we seek blending functions for computer aided geometric design, we should look to classical discrete probability theory. One standard way to generate discrete probability distributions is to use urn models. We shall describe this technique in the following section.

3. Friedman's Urn Model

Friedman [1] introduced the simple urn model described here. Consider an urn initially containing w white balls and b black balls. One ball at a time is drawn at random from the urn and its color inspected. It is then returned to the urn and a constant number c_1 of balls of the same color and a constant number c_2 of balls of the opposite color are added to the urn.

We now introduce the following notation:

$t = \dfrac{w}{w+b}$ = probability of selecting a white ball on the very first trial

$a_1 = \dfrac{c_1}{w+b}$ = percentage of balls of the same color added to the urn after the very first trial

$a_2 = \dfrac{c_2}{w+b}$ = percentage of balls of the opposite color added to the urn after the very first trial

$D_k^N(t) = D_k^N(a_1,a_2,t)$ = probability of selecting exactly k white balls in the first N trials given the initial conditions a_1,a_2,t (Here we treat a_1,a_2 as constants and t as a variable. Thus for distinct t — that is, for different w and b — we change c_1,c_2, to keep a_1,a_2 constant.)

$D(t) = D(a_1,a_2,t)$ = the probability distribution consisting of the functions $D_0^N(t),\ldots,D_N^N(t)$

$s_k^N(t) = s_k^N(a_1,a_2,t)$ = probability of selecting a white ball after selecting exactly k white balls in the first N trials

$f_k^N(t) = f_k^N(a_1,a_2,t)$ = probability of selecting a black ball after selecting exactly k white balls in the first N trials

$P = (P_0,\ldots,P_N)$ = control points

$D[P](t) = \sum_k D_k^N(t)P_k$ = curve with control points P and blending functions D(t)

The functions $D_0^N(t),\ldots,D_N^N(t)$ form a discrete probability distribution because they describe the probabilities of N+1 mutually exclusive events, one of which must occur. Therefore

$$\sum_k D_k^N(t) = 1 \qquad 0 \le t \le 1$$

$$D_k^N(t) \ge 0 \qquad 0 \le t \le 1$$

Moreover, in order to select exactly k white balls in the first N+1 trials, either exactly k or exactly k-1 white balls must be selected in the first N trials. This simple observation leads directly to the basic recursion formula

$$D_k^{N+1}(t) = f_k^N(t)D_k^N(t) + s_{k-1}^N(t)D_{k-1}^N(t)$$

By considering the contents of the urn after the N^{th} trial, it is easy to find explicit expressions for the functions $f_k^N(t)$, $s_k^N(t)$. Indeed

$$s_k^N(t) = \frac{\text{number of white balls in the urn}}{\text{total number of all the balls in the urn}}$$

$$= \frac{w + kc_1 + (N-k)c_2}{w + b + N(c_1+c_2)}$$

Dividing numerator and denominator by w+b, we get

$$s_k^N(t) = \frac{t + ka_1 + (N-k)a_2}{1 + N(a_1 + a_2)}$$

Similarly

$$f_k^N(t) = \frac{1-t + (N-k)a_1 + ka_2}{1 + N(a_1 + a_2)}$$

Since by definition

$$D_0^1(t) = 1-t$$

$$D_1^1(t) = t$$

we can use the recursion formula to calculate $D_k^N(t)$ for any k,N. Moreover it follows by the recursion formula and induction on N that $D_k^N(t)$ is a degree N polynomial in t which depends only on a_1, a_2, t.

For each pair of constants (a_1, a_2), we get a different distribution D(t) and hence a different curve D[P](t). Thus by varying the scalars (a_1, a_2) we can alter the shape of the curve without moving the control points. This technique allows us to fine-tune the shape of the curve by adjusting some scalar parameters. In the next section we list some of the important common properties of the family of curves D[P](t).

4. Probability and Geometry

Below we summarize some of the important geometric properties common to the curves D[P](t) together with the probabilistic considerations which induce these geometric characteristics. A similar list exists for surfaces.

	Urn	Blending Functions	Curve
1.	Probability Distribution	$\Rightarrow \sum_k D_k^N(t) = 1$	$\Rightarrow$ Well-Defined $\Rightarrow$ Exactly Reproduces Points $\Rightarrow$ Translating the Control Points Translates the Curve
2.	Probability Distribution	$\Rightarrow D_k^N(t) \geq 0$	$\Rightarrow$ Convex Hull Property $D[P](t) \leq$ Convex Hull (P)
3.	Symmetry between white balls and black balls	$\Rightarrow D_k^N(t) = D_{N-k}^N(1-t)$	$\Rightarrow$ Symmetry $D[P_0,\ldots,P_N](1-t)=D[P_N,\ldots,P_0](t)$
4.	If there are no white (black) balls initially in the urn, then no white (black) balls can ever be selected (only if $a_2=0$)	$\Rightarrow D_k^N(0) = 0 \quad k \neq 0$ $= 1 \quad k = 0$ $D_k^N(1) = 0 \quad k \neq N$ $= 1 \quad k = N$	$\Rightarrow$ Interpolates the First and Last Control Points $D[P](0) = P_0$ $D[P](1) = P_N$
5.	Expectation $= Nt$ (only if $a_2=0$)	$\Rightarrow \sum_k kD_k^N(t) = Nt$	$\Rightarrow$ Exactly Reproduces Straight Lines
6.	Relationship between first N and first N+1 picks	$\Rightarrow$ Recursion Formula $D_k^{N+1}(t) = f_k^N(t)D_k^N(t)+s_{k-1}^N(t)D_{k-1}^N(t)$	$\Rightarrow$ Geometric Construction Algorithms
7.	Counting (only if $a_2=0$)	$\Rightarrow$ Explicit Formula [2]	$\Rightarrow$ Polynomial Curve (also follows from recursion)
8.	Counting (only if $a_2=0$)	$\Rightarrow$ Formula for Raising Degree	$\Rightarrow$ Algorithm for Adding Control Points

Formula for item 7:

$$D_k^N(t) = \binom{N}{k} \frac{(t)\ldots(t+[k-1]a_1)(1-t)\ldots(1-t+[N-k-1]a_1)}{(1+a_1)\ldots(1+[N-1]a_1)}$$

Formula for item 8:

$$D_k^N(t) = \frac{(N+1-k)}{(N+1)}D_k^{N+1}(t) + \frac{(k+1)}{(N+1)}D_{k+1}^{N+1}(t)$$

[Notice that this formula is independent of a_1.]

	Urn		Blending Functions		Curve
9.	Two Urns Each with Two Colors	$\Longrightarrow$	$D^{MN}_{jk}(s,t)$ = probability of selecting exactly j white balls in the first M trials from urn 1, and exactly k white balls in the first N trials from urn 2 $= D^M_j(s)D^N_k(t)$	$\Longrightarrow$	Rectangular (Tensor Product) Surfaces
	One Urn with Three Colors	$\Longrightarrow$	$D^N_{jk}(s,t)$ = probability of selecting exactly j red balls and k white balls in the first N trials	$\Longrightarrow$	Triangular Surfaces
10.	? (Descartes' Original Law of Signs)	$\Longrightarrow$	Descartes' Law of Signs	$\Longrightarrow$	Variation Diminishing
11.	? (Descartes' Law of Signs)	$\Longrightarrow$	Linear Independence	$\Longrightarrow$	Non-Degenerate
				$\Longrightarrow$	Uniqueness
12.	? (Linear Independence)	$\Longrightarrow$	Polynomial Basis	$\Longrightarrow$	Subdivision Algorithm

It is clear from this summary that most of the important properties of the blending functions are grounded in their probabilistic interpretation. However, as yet, there exists no probabilistic explanation of Descartes' Law of Signs. Nevertheless, Descartes' Law of Signs has been shown to hold for the special cases $a_2 = 0$ and $a_1 = 0$, $a_2 = 1$. We conjecture that it is generally true for all the probability distributions which arise from Friedman's urn model (that is, for all values of a_1, a_2), but we know of no proof, probabilistic or otherwise, for this general conjecture.

5. Some Important Examples

a. Bezier Curves

When $a_1 = a_2 = 0$, the distribution $D(t)$ is the binomial distribution (sampling with replacement), and therefore the curve $D[P](t)$ is simply the Bezier curve with control points P.

b. Polya Curves

When $a_1 \neq 0$, $a_2 = 0$, the distribution $D(t)$ is the classical Polya distribution [3]. Therefore we shall call the curves $D[P](t)$ Polya curves. The Polya curves are simple generalizations of the Bezier curves and they share many of the same standard geometric properties (see section 4). To understand the geometric significance of the free parameter a_1, notice that

$$\begin{aligned} \lim_{a_1 \to \infty} D_k^N(t) &= D_0^1(t) = 1-t && k = 0 \\ &= 0 && k \neq 0,N \\ &= D_1^1(t) = t && k = N \end{aligned}$$

since if we add an infinite number of balls of the same color, we will always, with probability 1, select the same color ball as on the first trial. Therefore

$$\lim_{a_1 \to \infty} D[P](t) = (1-t)P_0 + tP_N$$

Thus a_1 controls the relative flatness of the Polya curve. By varying this scalar parameter, a designer could regulate the relative flatness of his curve without altering his control points.

c. B-Splines

When $a_1 = 0$, $a_2 = 1$, the distribution $D(t)$ actually generates the normalized uniform B-spline basis functions. To see why, consider 2 urns, one containing h white balls and 0 black balls, the other containing h black balls and 0 white balls. After one pick, the contents of the two urns are identical (see fig. 1).

Therefore, clearly,

$$D_k^N(1) = D_{k-1}^N(0)$$

Hence we can define a polynomial spline $S(t)$ by setting

$$\begin{aligned} S(t) &= D_{N-k}^N(t-k) && k \leq t \leq k+1 \\ &= 0 && t \leq 0 \text{ or } t \geq N+1 \end{aligned}$$

(See fig. 2.) By our previous argument these functions fit together continuously. Moreover, by a simple inductive argument involving the recursion formula, these polynomials actually fit together smoothly up to order N-1.

$$\int_0^{N+1} S(t)dt = \sum_k \int_0^1 D_k^N(t)dt = 1$$

It follows that S(t) is the normalized uniform B-spline basis function with knot vector (0,1,...,N+1); thus the curve D[P](t) really represents one segment of a typical B-spline curve.

When $a_1 = 0$, $a_2 = 1$, the recursion formula reduces to

$$D_k^{N+1}(t) = \frac{(1 - t + k)\, D_k^N(t)}{(1 + N)} + \frac{(t + N + 1 - k)\, D_{k-1}^N(t)}{(1 + N)}$$

This formula is actually the Cox/De Boor recursion formula in its simplest form. Indeed, the general Cox/De Boor formula is a simple consequence of this identity. Simple urn models which generate the uniform B-spline basis functions with multiple knots are also known, and as would be expected these urn models are just simple generalizations of the one discussed here.

d. Generalized Splines

When $a_2 = 1+a_1$, the distribution D(t) generates a generalized spline function. Indeed in this case it is again easy to show that

$$D_k^N(1) = D_{k-1}^N(0)$$

Therefore the spline function

$$\begin{aligned} S(t) &= D_{N-k}^N(t-k) && k \le t \le k+1 \\ &= 0 && t \le 0 \text{ or } t \ge N+1 \end{aligned}$$

is, in general, continuous, though not necessarily differentiable. These distributions D(t) have much in common with each other and with the standard B-spline distribution. For example, if $s_N(t)$ is the a priori probability of selecting a white ball on the N^{th} trial, it has been shown that if $a_2=1+a_1$, then

$$\begin{aligned} s_N(t) &= t && N = 1 \\ &= 1/2 && N > 1 \end{aligned}$$

It follows immediately therefore that the expectation is simply

$$\sum k\, D_k^N(t) = \sum s_k(t) = t + \frac{(N-1)}{2}$$

Thus the distributions D(t) for which $a_2 = 1+a_1$ are natural generalizations of the B-spline distribution, and the curves D[P](t) are generalizations of the B-spline curves in much the same way that the Polya curves are generalizations of the Bezier curves.

6. Conclusion

In this paper we have tried to elaborate upon the deep connection between probability theory and computer-aided geometric design. Urn models are a powerful tool for generating discrete probability distributions, and built into these special distributions are many propitious properties essential to the blending functions of computer-aided geometric design. This fact allows us to use probabilistic arguments to simplify, unify, and generalize many geometric results. We believe that this link between probability and geometry will ultimately prove beneficial to both disciplines, and we expect that it will continue to be a productive area for future inspiration and research.

REFERENCES

1. Friedman, B., A Simple Urn Model, Comm. Pure Appl. Math. 2, 59-70, 1949.

2. Chung, K. L., Elementary Probability Theory with Stochastic Processes, Springer-Verlag, New York, 1975.

3. Polya, G., Sur quelques points de la théorie des probabilitées, Ann. Inst. H. Poincaré 1, 117-161, 1931.

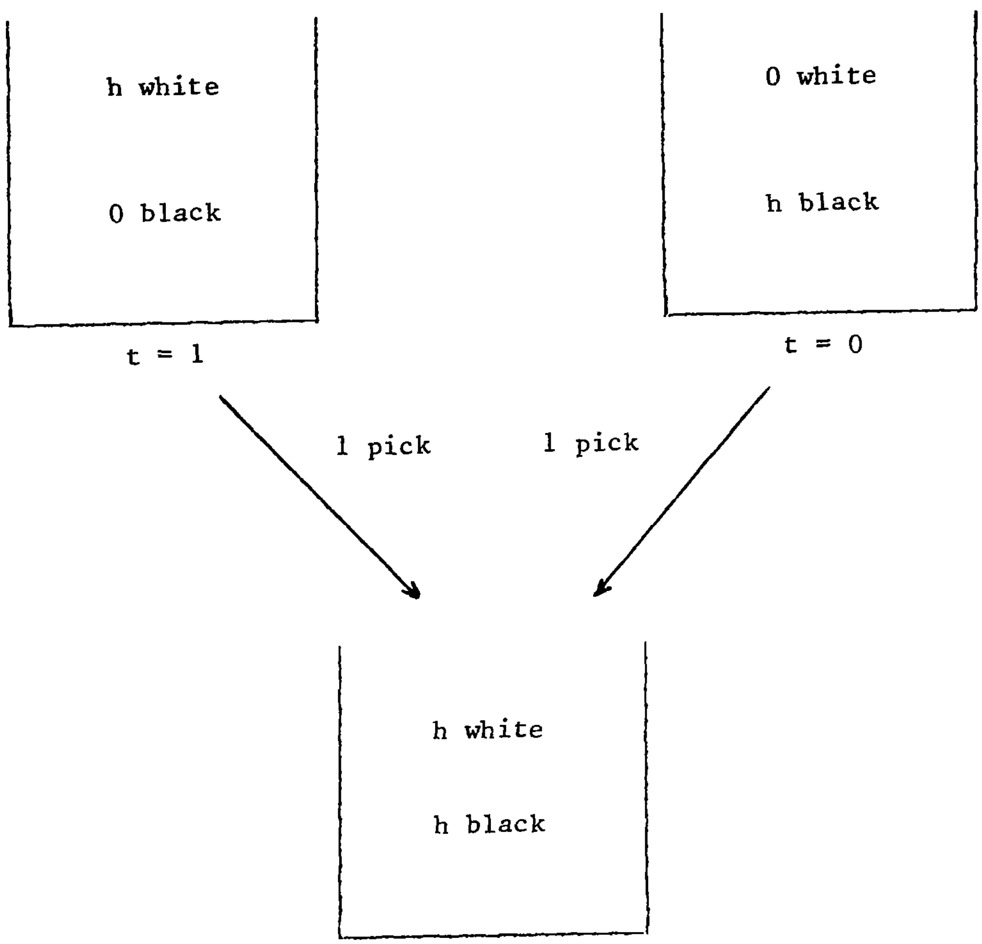


Figure 1.- $D_k^N(1) = D_{k-1}^N(0)$.

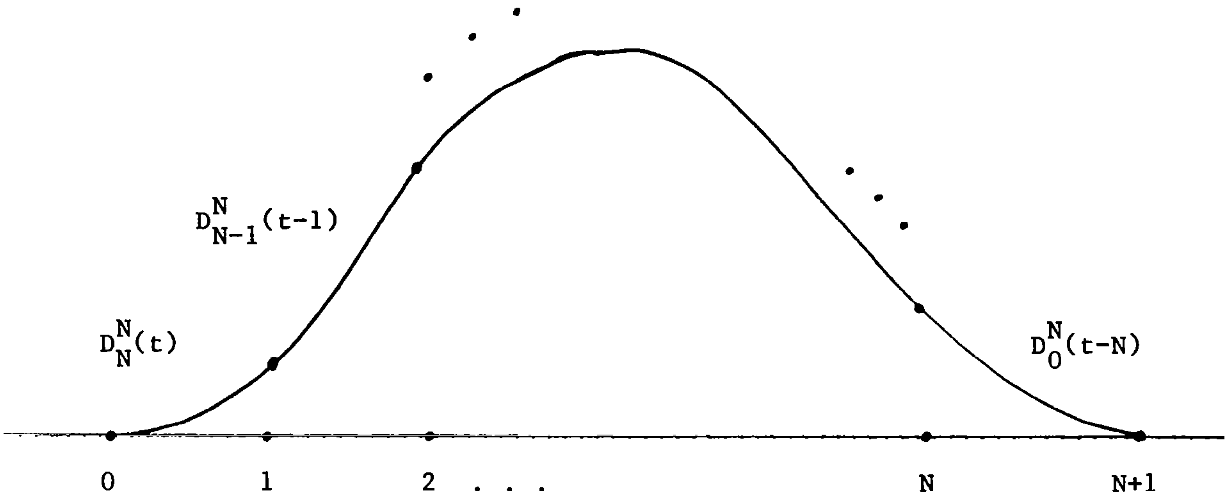


Figure 2.- The spline S(t).

INTERSECTION OF PARAMETRIC SURFACES USING LOOKUP TABLES

Samir L. Hanna, John F. Abel,
and Donald P. Greenberg
Department of Structural Engineering
and
Program of Computer Graphics
Cornell University
Ithaca, N.Y.

When primitive structures in the form of parametric surfaces are combined and modified interactively to form complex intersecting surfaces, it becomes important to find the curves of intersection. One must distinguish between finding the "shape" of the intersection curve, which may only be useful for display purposes, and finding an accurate mathematical representation of the curve, which is important for any meaningful geometric modeling, analysis, design, or manufacturing involving the intersection.

The intersection curve between two or more parametric surfaces is important in a variety of computer-aided design and manufacture areas. A few examples are shape design, analysis of groins, design of fillets, and computation of numerically controlled tooling paths. The algorithm presented here provides a mathematical representation of the intersection curve to a specified accuracy. It also provides the database that can simplify operations such as hidden-surface removal, surface rendering, profile identification, and interference or clearance computations.

Unless the two surfaces are planes, the intersection curve between them is determined by the solution of nonlinear equations. The calculation of the intersection curve may be regarded either as a problem involving the solution of simultaneous nonlinear equations, or as a minimization problem in which the squared distance between variable points on each surface is minimized. These methods are computationally expensive and have been avoided in this development.

The algorithm presented here is a "divide and conquer" algorithm where the number of subdivisions depends on the accuracy required. It provides data that can be used not only for display purposes but also for analysis and design. In general terms, the algorithm consists of the following steps:

1. Separability test: Using the coarse mesh, one builds a list of the interfering polygon pair numbers on the different surfaces. From that list, the bounds of all of these polygons on each surface are found (intersection bounds).

2. Setup of lookup tables: Within the intersection bounds of each surface a fine mesh (lookup table) is generated. The resolution of this mesh depends upon the required accuracy.

3. Tracing the intersection curve: The lookup tables of each pair of interfering patches are scanned for point pairs that are apart a distance less than or equal to a precomputed tolerance. These point pairs are referred to as "neighboring pairs." The neighboring pairs lie in the vicinity of the intersection curve and not exactly on it.

4. Refining and connecting the intersection curve: A better estimate is computed for the neighboring pairs by the intersection of the planes tangent and normal to the surface at the neighboring points. The refined points are then connected to form the intersection curve.

Besides using the lookup tables for tracing the intersection curve, the tables are employed in simple real-time mesh editing procedures for finite element preprocessing, as nodes and lines may be "dragged" or "added" with only lookup operations performed and barely any computation is needed. The same type of tables can be used for silhouette tracing, hidden-surface removal, surface rendering, and interferance or clearance computations.

Figure 1 distills the basis for the divide and conquer approach. Figures 2 through 9 illustrate various steps of the algorithm for a selected simple configuration--two quarter cylinders. As the additional examples in Figures 10-11 indicate, however, the procedure is clearly applicable to more complex configurations. In fact, combinations of three different types of parametric surfaces are handled in the present implementation: spline-lofted, rotational, and translational surfaces.

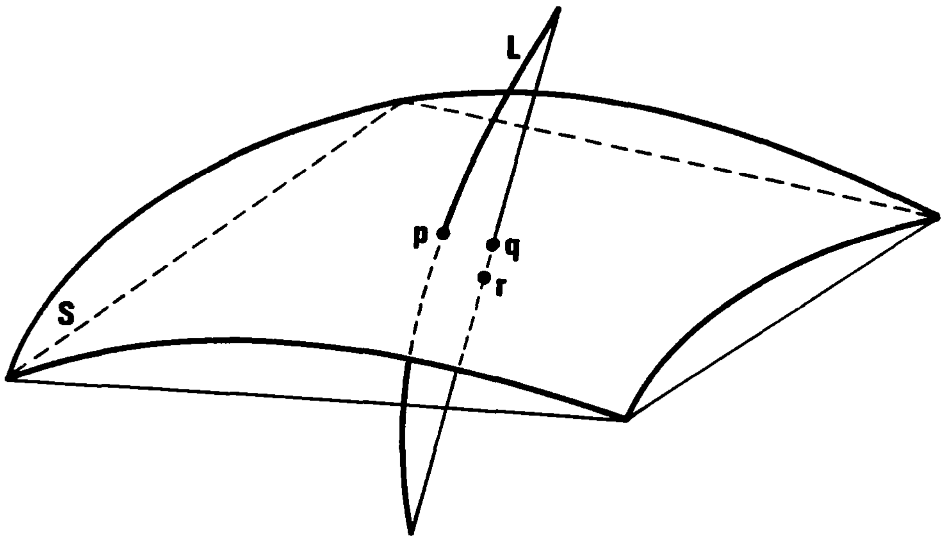


Figure 1. Intersection of a curve L and a surface S. The points p, q and r approach a single point as the resolution of subdivision of curves and surfaces is increased.

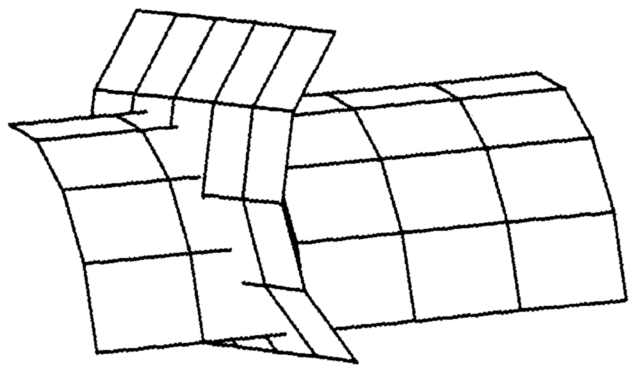


Figure 2. To show the configuration of the illustrative problem, a coarse hidden-line view is shown prior to the solution for the intersection curve of these two quarter cylinders.

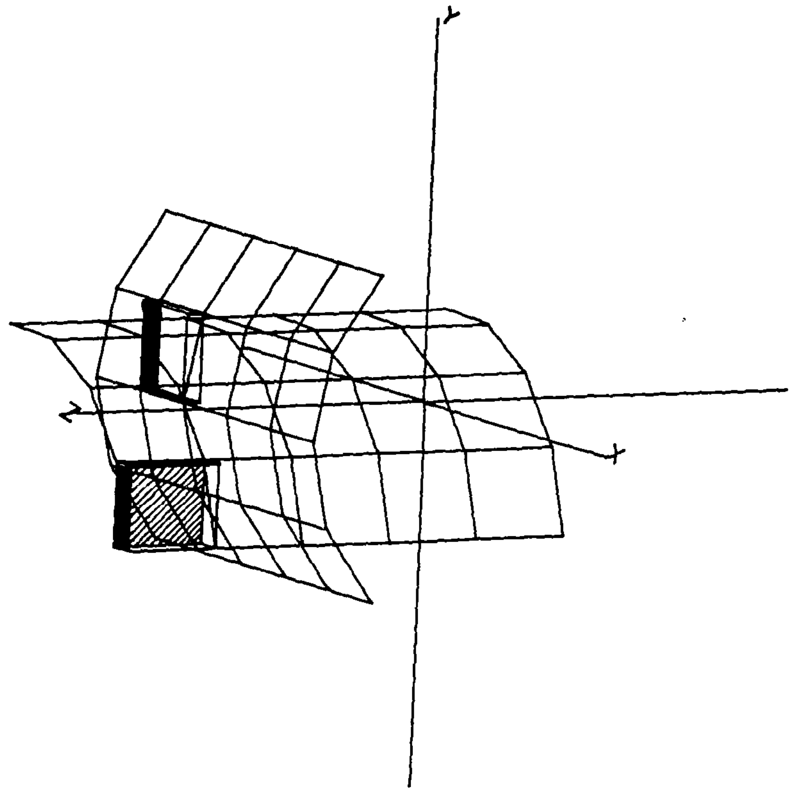


Figure 3. A separability test being performed for a pair of min-max enclosing boxes. Such boxes may enclose single polygons or groups of polygons.

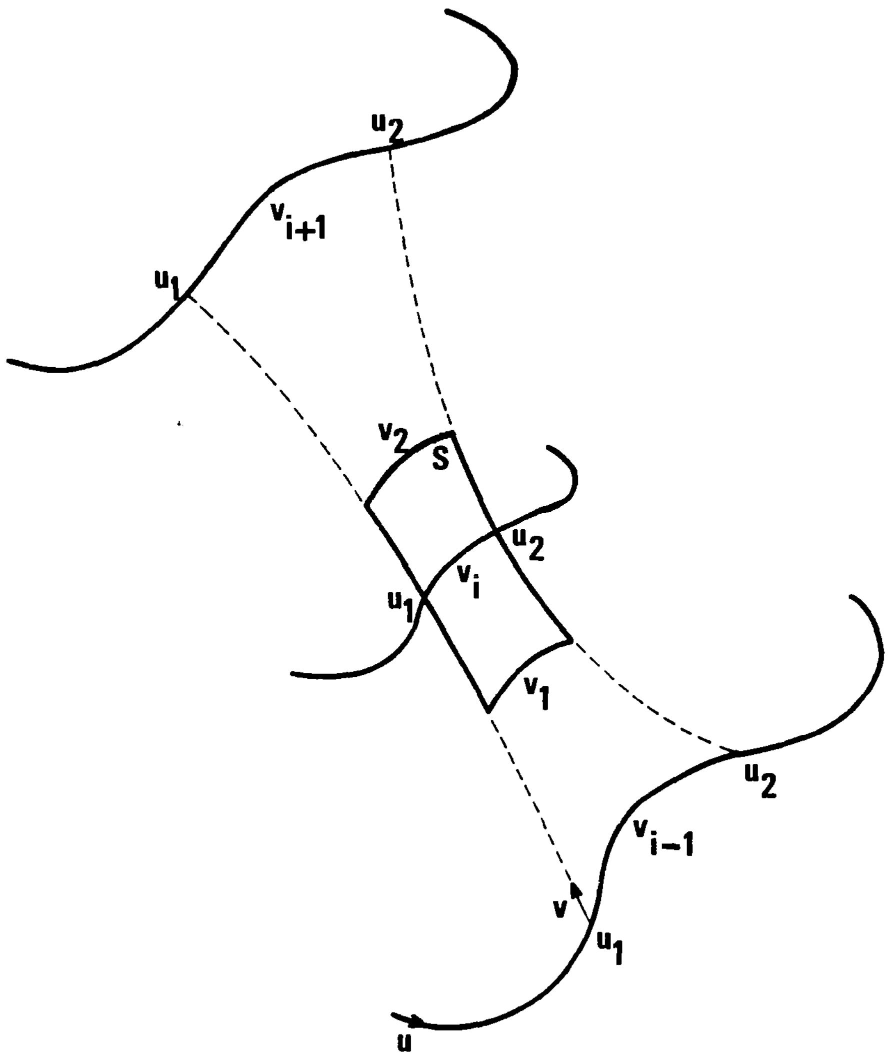


Figure 4. Portion of a spline-lofted surface within intersection bounds. An efficient difference algorithm permits generation of lookup tables for just this portion of the surface.

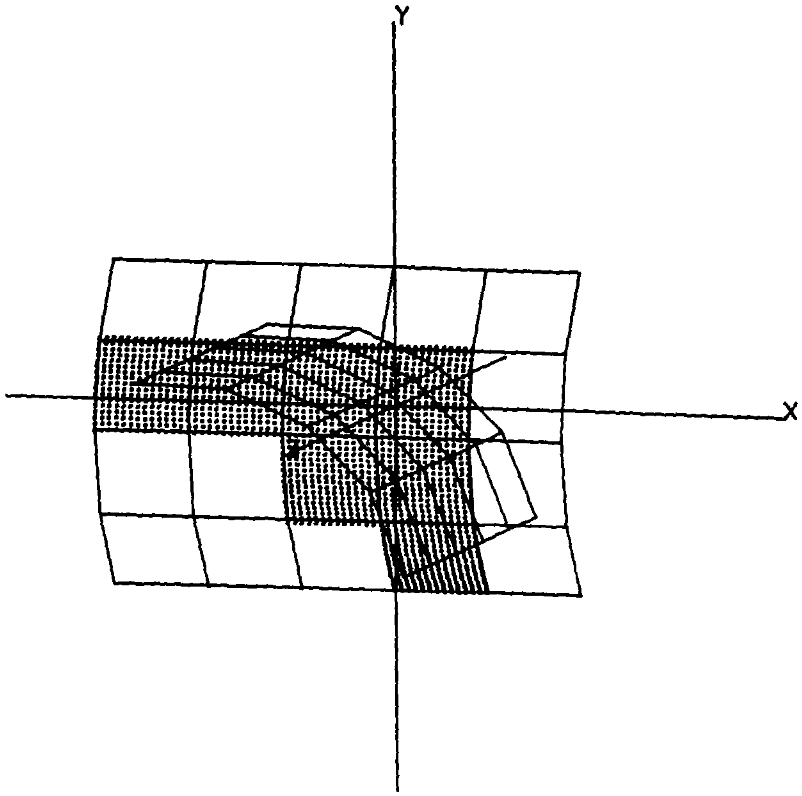


Figure 5. The lookup table for one of the quarter cylinders.

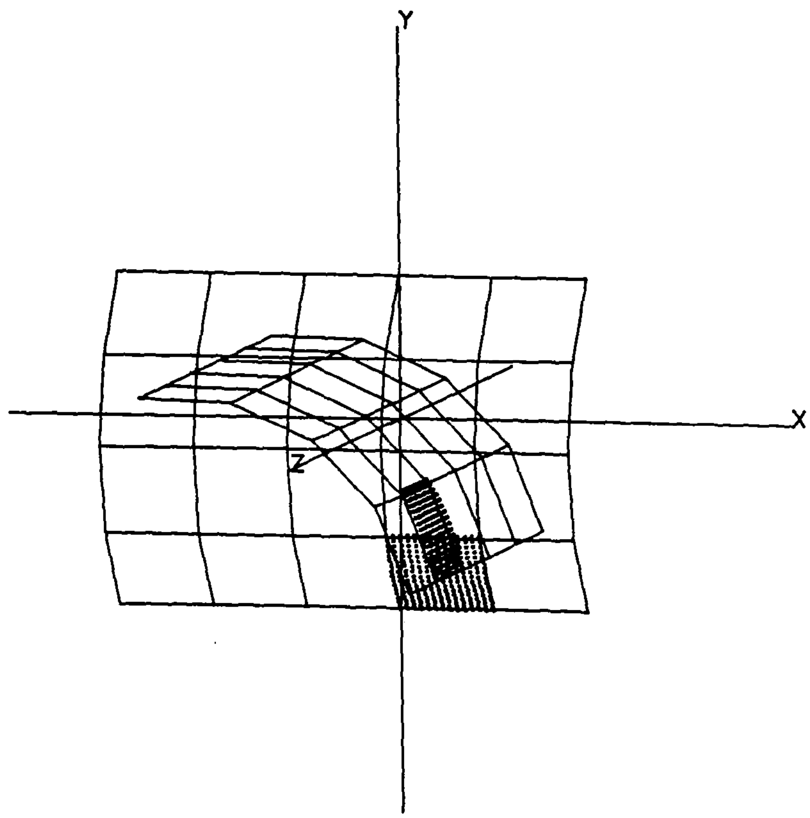


Figure 6. Lookup table points on each surface within a pair of interfering patches.

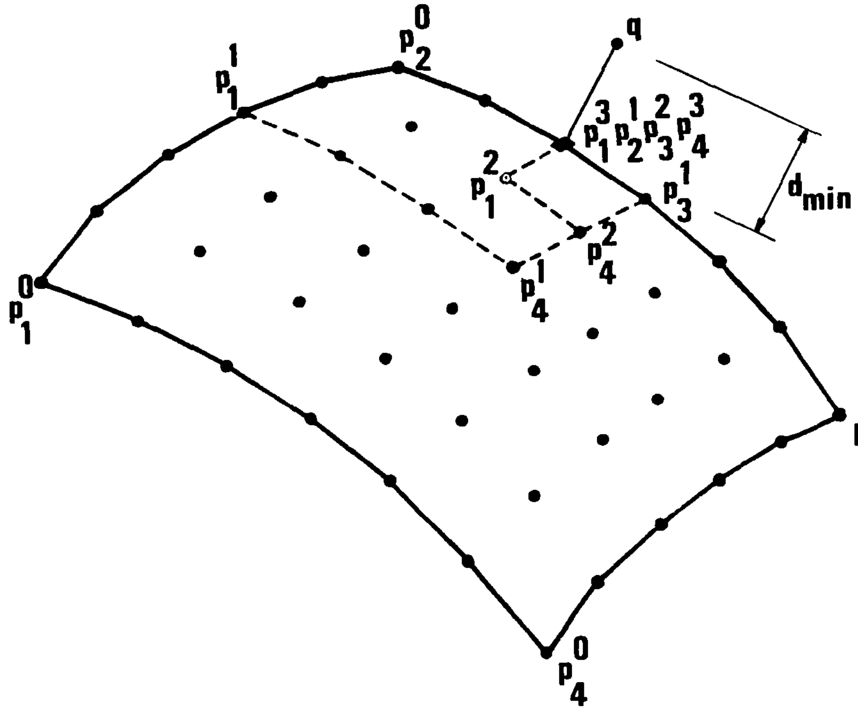


Figure 7. The minimum distance between the points within the patch $p_1^0 p_2^0 p_3^0 p_4^0$ and the point q is found by bisection on the lookup table points within the patch.

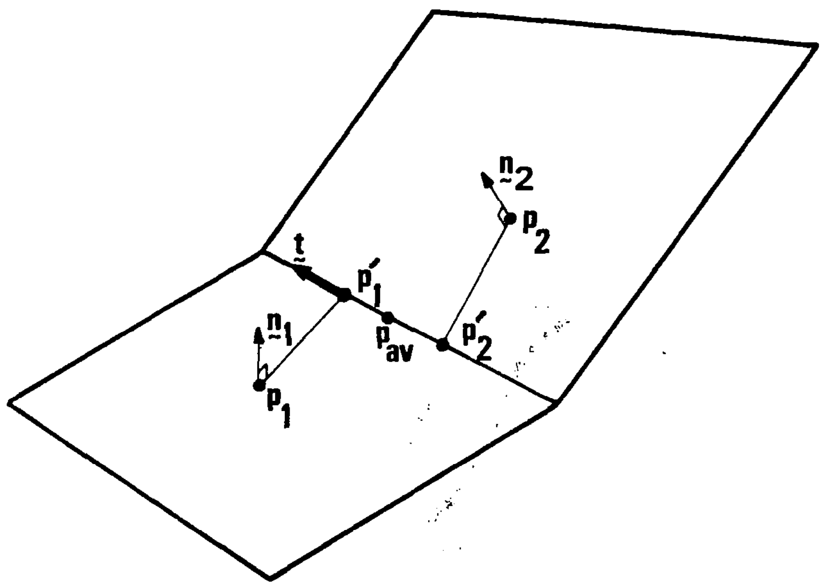


Figure 8. Refinement of intersection by finding p_{av} from the neighboring pair p_1 and p_2.

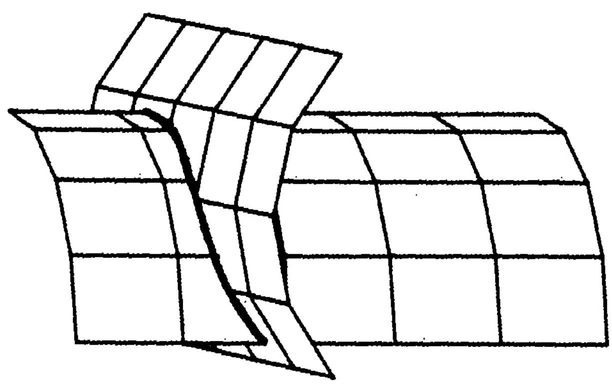


Figure 9. The intersection curve of the two quarter cylinders shown (without hidden portions removed) superimposed on the hidden-line view of Figure 2.

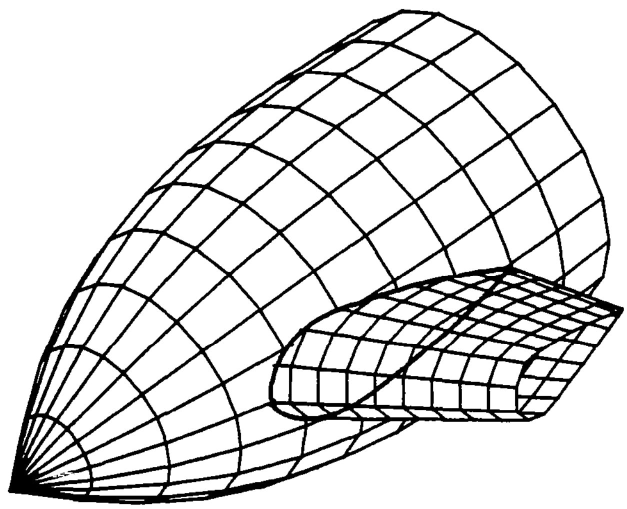


Figure 10. The intersection curve of two surfaces with one having a slope discontinuity (without hidden portions removed) superimposed on a hidden-line view of the two surfaces.

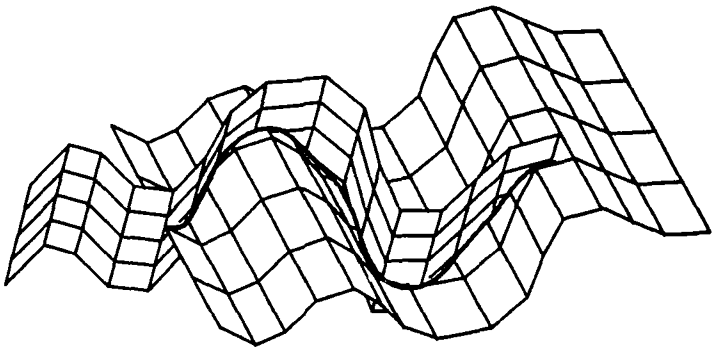


Figure 11. The intersection of two nonuniformly corrugated sheets. The intersection curve is superimposed without removing the hidden portions on a hidden-line view of the two surfaces.

SURFACE GRID GENERATION FOR WING-FUSELAGE BODIES

R. E. Smith, R. A. Kudlinski, and J. I. Pitts
NASA Langley Research Center
Hampton, Virginia

EXTENDED ABSTRACT

In the application of finite-difference methods to obtain numerical solutions of viscous compressible fluid flow about wing-fuselage bodies (fig. 1), it is advantageous to transform the governing equations to an idealized boundary-fitted coordinate system (ref. 1). The advantages are reduced computational complexity and added accuracy in the application of boundary conditions. The solution process requires that a grid be superimposed on the physical solution domain which corresponds to a uniform grid on a rectangular computational domain (uniform rectangular parallelepiped, fig. 2).

Grid generation is the determination of a "one to one" relationship between grid points in the physical domain and grid points in the computational domain. Two approaches to grid generation -- algebraic methods (refs. 2, 3 and 4) and elliptic systems methods (refs. 5, 6 and 7) -- depend on physical exterior boundary grids to compute interior grids. Computing boundary grids on curved intersecting boundary surfaces is itself a complex task with both geometric and grid spacing control constraints.

For flow about a wing-fuselage body, the physical domain is the region between the wing-fuselage surface and an outer boundary surface where free-stream conditions are assumed to exist. Using a boundary-fitted coordinate system, the physical grid must conform to the physical boundary geometry and be concentrated in regions where large gradients are anticipated. The first step in obtaining a grid on this region is the computation of a grid on the wing-fuselage surface which maps onto two sides of the computational grid (fig. 3). This presentation describes a technique for computing wing-fuselage surface grids using the Harris geometry (refs. 8 and 9) and the software for smooth-surface representation presented in reference 9. Grid spacing control concepts which govern the relationship between the wing-fuselage surface and the computational grid are also presented.

Each component of an aircraft is described by the Harris geometry in terms of cross sections. A fuselage is described by cross sections along the x-body axis (fig. 4) and a wing is described as airfoil sections in the spanwise z-direction (fig. 5). In turn, each airfoil section is defined by the coordinates of its leading-edge position, its chord length, and its camber line and Δy coordinates. In reference 9 cubic splines are fit along and across the cross-sectional data for each component. The position and derivative data from the spline fits provide the corner parameters for a Coons' patch surface definition (ref. 10).

The transformation of the wing surface and fuselage surface to adjoining planes in the computational grid requires the determination of the intersection curve between the two physical surfaces. This is accomplished by using the capability described in reference 9 for computing the intersection of an arbitrarily oriented plane and the aircraft surface components. The plane-component intersection is

obtained from the simultaneous solution of the equation of a plane and the accumulation of Coons' patch equations for each component. The results are three-dimensional curves in a plane. A plane is described by three unique points, and maintaining the same x-coordinate for all three points describes planes perpendicular to the x-body axis. Varying the x-coordinate in the intersection option presented in reference 9 produces interpolated cross sections of the wing and fuselage (fig. 6). Within a plane the intersections of the curves from each component are computed by an iterative search (fig. 7). The intersection points along with the x-coordinate defining the plane denote points on the wing-fuselage intersection curve. The x-coordinates for which there is only one planar curve intersection point denote the leading and trailing points on the wing-fuselage intersection curve.

In order to simplify the description of the surface grid generation, the wing-fuselage grid is divided into an upper part and a lower part. The grid behind the wing can be generated and coupled with the upper and lower front grids to map onto a single rectangular parallelepiped. The remaining discussion, however, will deal only with the upper wing-fuselage grid.

The next step in the surface grid generation process is to relate a computational coordinate (ξ) to the curve defined by the intersection of the wing and fuselage. The ξ-coordinate is uniformly discretized on the unit interval $0 \leq \xi \leq 1$. This coordinate is related with a "one to one" function to the normalized approximate arc length along the intersection curve. The normalization factor is the maximum value of the approximate arc length. The physical coordinates of the intersection curve are functions of the approximate arc length, and through the relationship of the approximate arc length to the computational coordinate the intersection curve is therefore a function of the computational coordinate. In accordance with the techniques described in references 3, 4 and 11 the grid spacing in the physical coordinates is controlled by the single-valued function relating the computational coordinate to the arc length variable. A low slope in this function corresponds to a physical grid concentration and a high slope corresponds to a dispersion in the physical grid (fig. 8). A concentration of grid points near the leading edge of the wing-fuselage junction is obtained with an exponential function and is demonstrated in the example to be presented.

With the distribution of grid points along the wing-fuselage intersection curve, a corresponding grid point distribution along the terminal boundary of the fuselage surface is required. We assume that the terminal fuselage surface boundary is along the top of the fuselage with the z-coordinate equal to zero (fig. 9). The η computational coordinate is equal to one, whereas it is equal to zero for the wing-fuselage intersection. A distribution of grid points along the terminal curve and the creation of a relation to the ξ computational coordinate are obtained in a manner similar to that for the wing-fuselage intersection curve. Using corresponding grid points on the two boundary curves ($\eta = 0$, $\eta = 1$, $0 \leq \xi \leq 1$), grid curves are computed on the fuselage surface. This is again done with the Coons' patch surface representation and plane intersection capability. Planes are defined by three points, two of which are on the boundary curves. The third point is defined to be on the x-body axis with the x-coordinate equal to the x-coordinate of the wing-fuselage grid point (fig. 10). Grid curves are computed for each pair of points on the two boundaries, and a distribution of grid points along the curves with corresponding computational points in the η-direction is computed in a similar manner to the previously described distributions. A fuselage grid is shown in figure 11 with grid concentration toward the wing leading edge and the wing-fuselage intersection curve.

The remaining problem for the wing-fuselage surface grid generation is the computation of the grid on the wing surface. The available data include the wing-fuselage intersection grid points and the Coons' patch representation of the wing surface. A redefinition of the surface is performed using bicubic splines with arc length parameterizations corresponding to the ξ-variable in the chordwise direction and the ζ computational coordinate in the spanwise direction along the wing. Grid spacing control is applied through single-valued functions between the computational and the arc length variables. A combined wing-fuselage surface grid is shown in figure 12.

Surface grid generation is a critical element in the computation of three-dimensional flow field grids. For the computation of wing-fuselage surface grids, a procedure using existing, readily available surface generation software has been outlined. A means of relating computational coordinates to surface grids through arc length parameterization has been presented. Grid spacing control has been applied with a relation between the arc length variables and the computational coordinates. It is anticipated that further improvements will be made for automated or interactive surface grid computation.

REFERENCES

1. Smith, R. E.: Two Boundary Grid Generation for the Solution of the Three-Dimensional Navier-Stokes Equations. NASA TM-83123, May 1981.

2. Smith, R. E.: Algebraic Grid Generation, Numerical Grid Generation. Elsevier Sciences Publishing Co., Inc., New York, NY, 1982.

3. Eiseman, P. R.; and Smith, R. E.: Mesh Generation Using Algebraic Techniques. Numerical Grid Generation Techniques, NASA CP-2166, 1980, pp. 73-120.

4. Smith, R. E.; and Weigel, B. L.: Analytical and Approximate Boundary-Fitted Coordinate Systems for Fluid Flow Simulation. AIAA Paper 80-0192, AIAA 18th Aerospace Sciences Meeting, Pasadena, CA, Jan. 1980.

5. Thompson, J. F.; Thames, F. C.; and Mastin, C. W.: Boundary-Fitted Curvilinear Coordinate Systems for Solution of Partial Differential Equations on Field Containing Any Number of Arbitrary Two-Dimensional Bodies. NASA CR-2729, 1977.

6. Thompson, J. F.: Numerical Solution of Flow Problems Using Body-Fitted Coordinate Systems. Lecture Series in Computational Fluid Dynamics. (W. Kallmann Ed.) Hemisphere, 1980.

7. Thompson, J. F.; and Mastin, C. W.: Grid Generation Using Differential Systems Techniques. Numerical Grid Generation Techniques, NASA CP-2166, 1980, pp. 37-72.

8. Craidon, C. B.: Description of a Digital Computer Program for Airplane Configuration Plots. NASA TM X-2074, 1970.

9. Craidon, C. B.: A Computer Program for Fitting Smooth Surfaces to an Aircraft Configuration and Other Three-Dimensional Geometries. NASA TM X-3206, 1975.

10. Coons, S. A.: Surfaces for Computer Aided Design of Space Forms. MAC-TR-41, U.S. Air Force, June 1967. (Available from NTIS as AD-663504.)

11. Smith, R. E.; Kudlinski, R. A.; and Everton, E. L.: A Grid Spacing Control Technique for Algebraic Grid Generation Methods. AIAA Paper 82-0226, AIAA 20th Aerospace Sciences Meeting, Orlando, FL Jan. 1982.

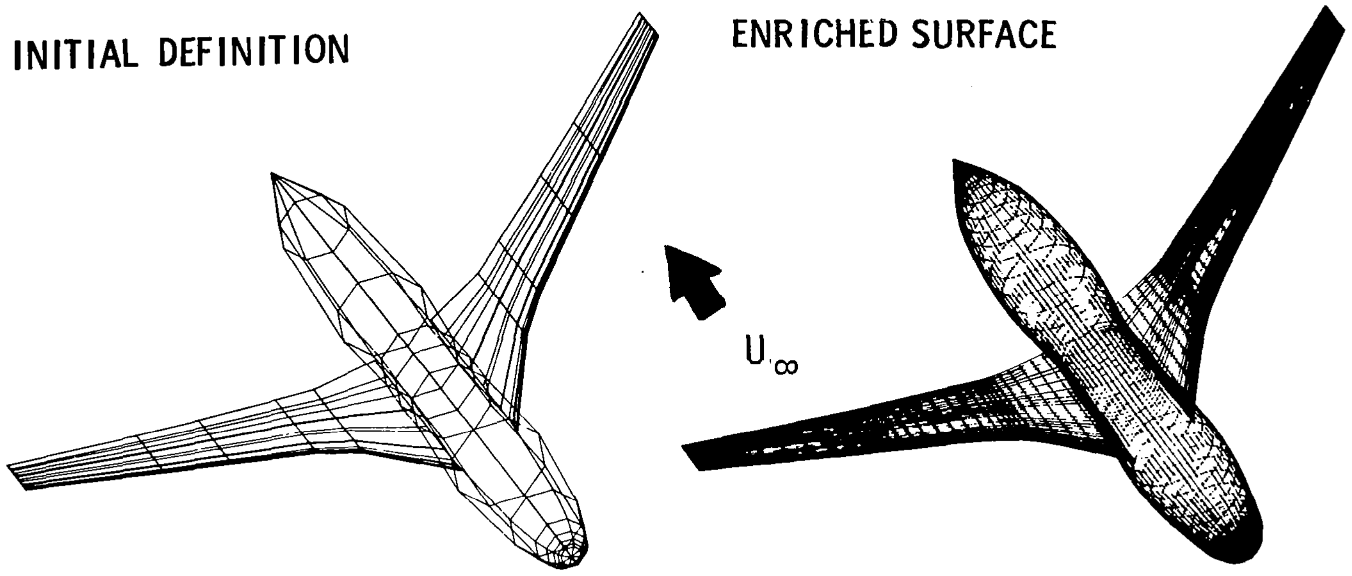


Figure 1.- Wing-fuselage combination for flow field boundaries.

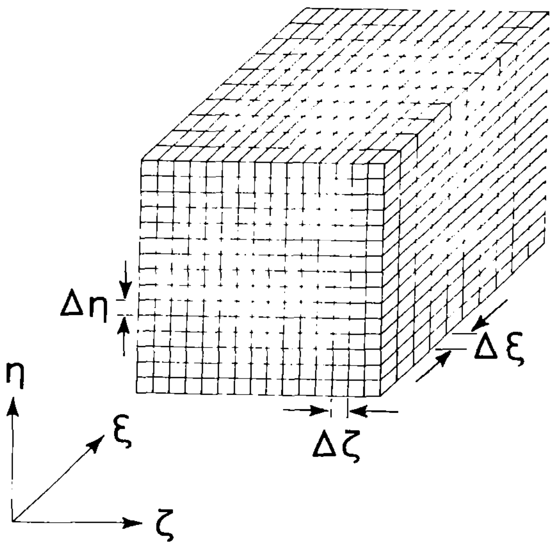


Figure 2.- Computational grid - rectangular parallelepiped.

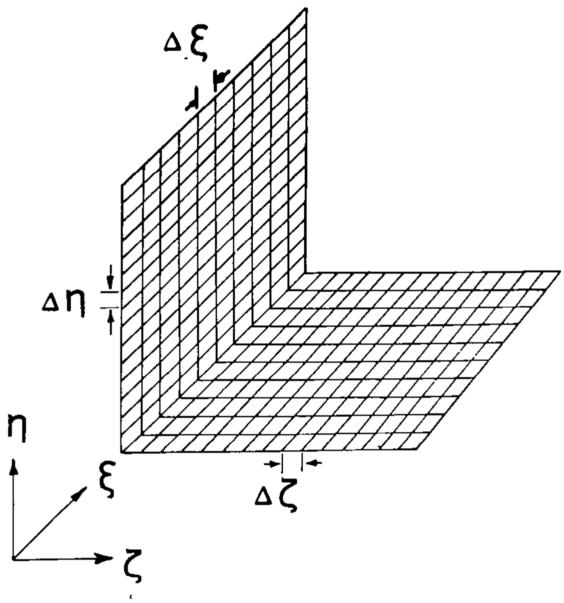


Figure 3.- Computational grid for a wing-fuselage surface.

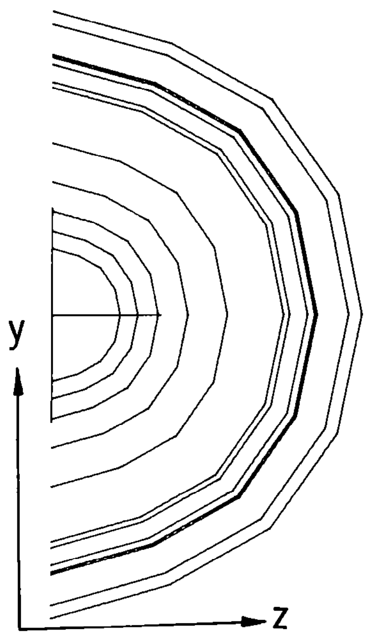


Figure 4.- Cross sections for fuselage description.

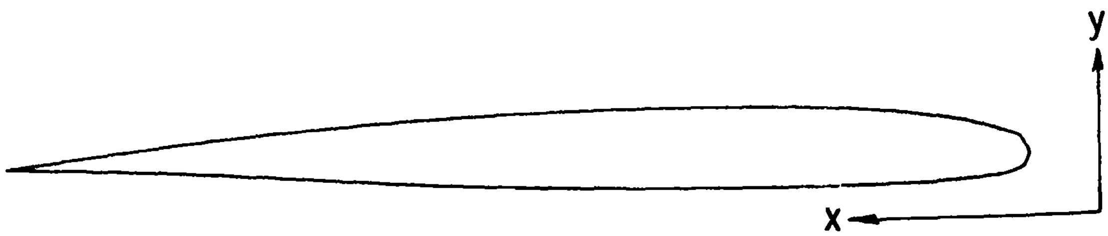


Figure 5.- Airfoil sections for wing description.

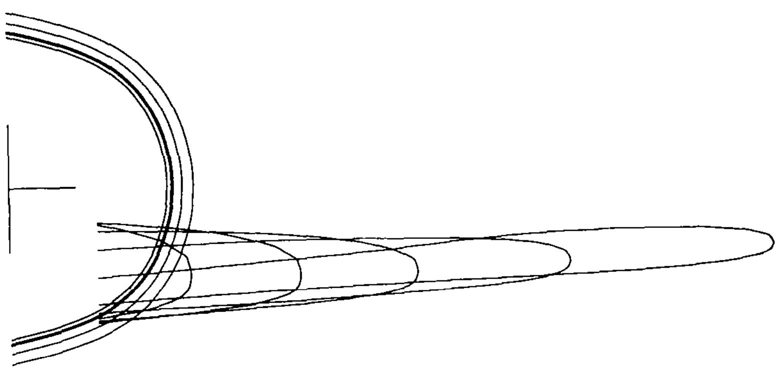


Figure 6.- Smooth cross sections of wing and fuselage.

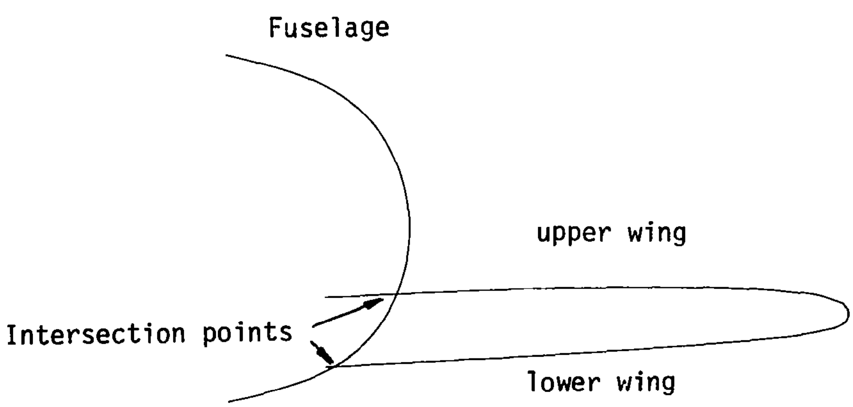


Figure 7.- Wing-fuselage intersection.

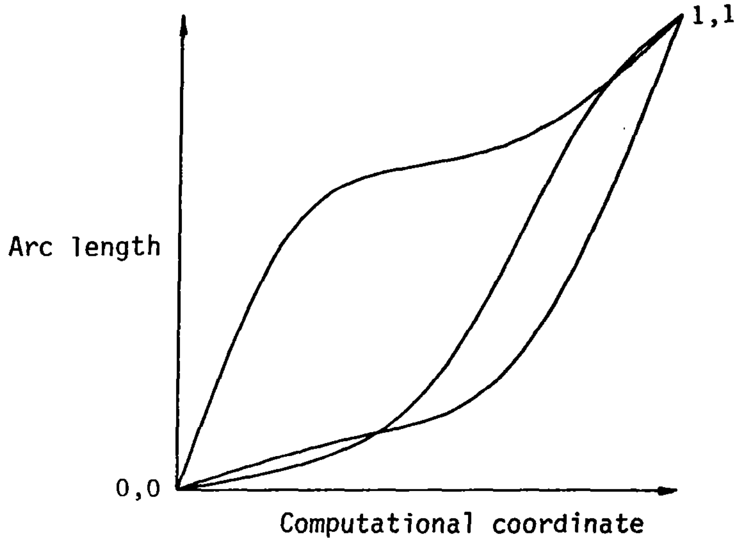


Figure 8.- Grid spacing control function.

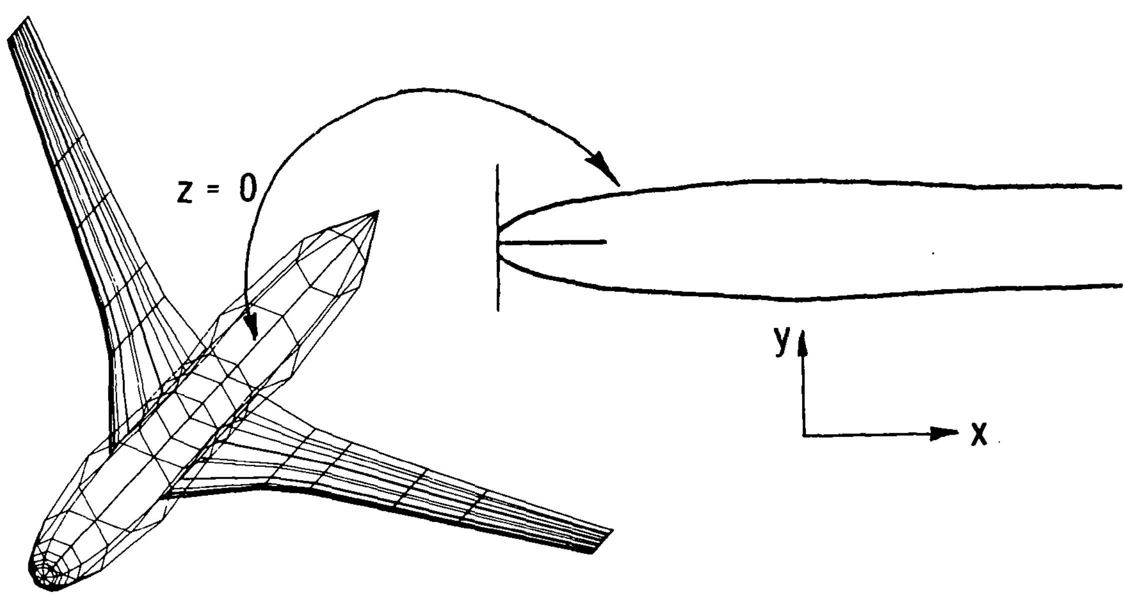


Figure 9.- Boundary definition in the η direction.

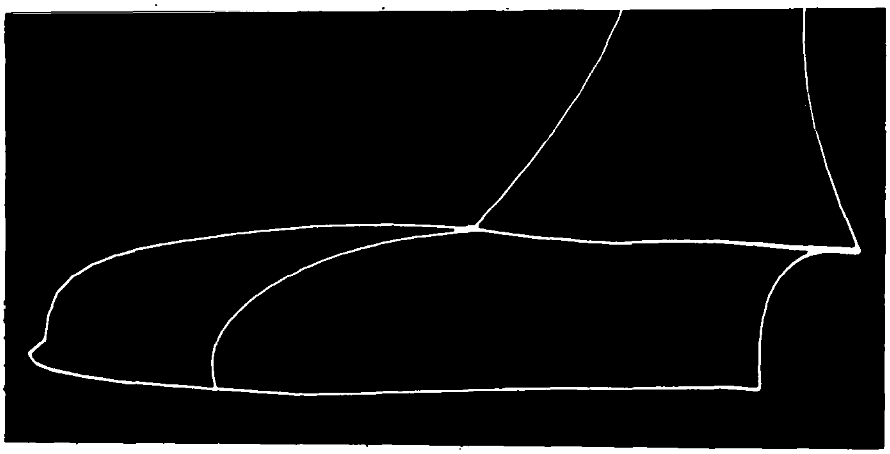


Figure 10.- Grid line definition on the fuselage.

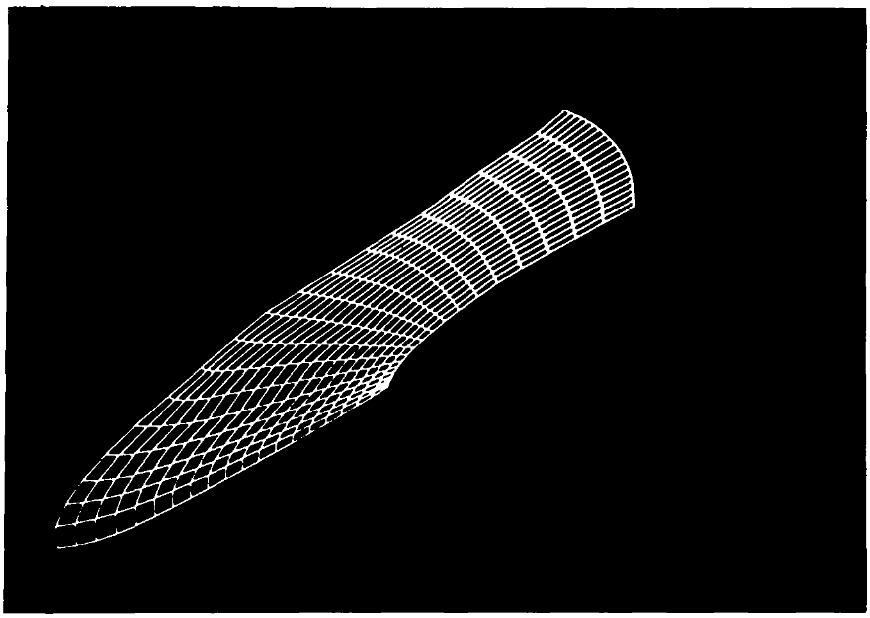


Figure 11.- Fuselage grid.

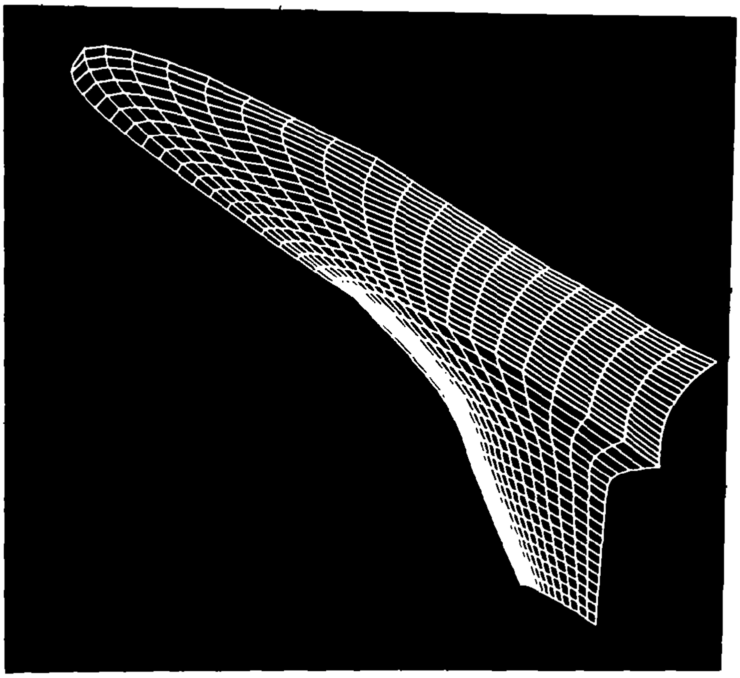


Figure 12.- Wing-fuselage grid.

SOLID MODELLING: HISTORY, STATUS, AND CURRENT RESEARCH DIRECTIONS*

Herbert B. Voelcker
Production Automation Project
University of Rochester
Rochester, New York

The term "solid modelling" encompasses an emerging body of theory, techniques, and systems focused on "informationally complete" representations of solids -- representations that permit (at least in principle) any well-defined geometrical property of any represented solid to be calculated automatically.

Research in solid modelling became visible in the mid-1960's and burgeoned in the 1970's. A first generation of experimental systems appeared in the early and mid-1970's and, while limited in many ways, elicited growing waves of interest and -- more importantly -- theoretical research. By the late 1970's enough was known about solid-modelling theory, fundamental algorithms, and certain key applications to enable a second generation of research systems to be built. This second generation has enough power to be industrially useful, and the vendors are moving swiftly to make available, in supported systems, the best of the now-stable and well understood late 1970's solid modelling technology. They are also interfacing their new solid modelling systems to their extant wireframe systems in order to exploit their many wireframe application packages.

Research aimed at producing new generations of solid modelling systems, and at extending the applications of solid modelling, is in progress and will be discussed briefly in this paper. Three major bodies of work are emerging: solid-modelling extensions, applications in design, and applications in production. Each area contains many parallel streams of work, with considerable cross-flow of concepts and techniques between the streams and areas.

Under 'extensions' one finds active research aimed at extending the domains of solid modelling systems to cover sculptured, blended, and toleranced objects. Improving the performance of systems is another active area, with effort focused on the search for new algorithms, improving known algorithms by exploiting spatial and representational locality, and -- increasingly -- the design and VLSI

----

*The Production Automation Project is supported by the National Science Foundation and by companies in the P.A.P.'s Industrial Associates Program.

implementation of "geometry engines" (special-purpose computers). Still other research is focused on improving human-user interfaces. This work may be viewed as a search for more congenial blends of graphical "poking" techniques and symbolic programming techniques. Finally there is continuing research on fundamental issues: approximation strategies, identification of new generic-problem algorithms, the effects of errors in numerical geometry, and so forth.

Design-applications research seeks mainly (at present) to equip solid modelling systems with very powerful analytical facilities, e.g. wholly or partially automated packages to do kinematic, dynamic, interference, and finite-element analysis. One line of work is aimed at interfacing available kinematical and dynamical packages (IMP, ADAMS, DRAM, ...) to modelling systems, and should attain its goals in the near future. Methods for doing static interference checking automatically are known and are available in some modelling systems, but dynamical interference analysis (e.g. on robot motions) is a largely open problem. In finite elements, current research is centered on developing automatic or almost automatic mesh generators to feed NASTRAN and similar analysis programs. Useful results should appear soon.

Research aimed at exploiting solid modelling to automate various production activities is rather limited at present, but it is likely to grow into the largest and most diverse area of all. The key to progress seems to lie in developing mathematical process models that are counterparts to, or complement, our powerful models for solid objects. Process models naturally are very process dependent, and thus one finds embryonic and largely independent streams of work aimed at modelling and analyzing machining processes, casting and molding processes, deformation processes, and so forth. (Mathematical process modelling has been going on for decades; the new element is the appearance of very powerful computational tools for "doing" 3-D solid geometry.) The research on machining and certain motional processes is likely to produce the earliest broadly useful results.

In summary, solid modelling is now an available and industrially viable technology whose importance is likely to grow dramatically in the remaining years of this century.

An extended and only slightly obsolescent version of this talk, and a useful 81-entry bibliography, may be found in "Solid Modelling: A Historical Summary and Contemporary Assessment", by A. A. G. Requicha and H. B. Voelcker, IEEE Computer Graphics and Applications, pp.9-24, March 1982.

History, Status, & Current Research Directions

Introduction

- The central role of geometry in discrete goods design and production
- Fatal deficiencies in traditional methods

Evolution of Solid Modelling

- Early work in the 1960's
- The 1970's: Golden Era 1
- Related streams of work

The Current (Commercial) Status

Contemporary Research

- Solid modelling extensions
- Applications in design
- Applications in production

Education

THE 1970'S: GOLDEN ERA 1

The First Generation of Systems

- SYNTHAVISION, SHAPES
- TIPS
- GEOMAP, GLIDE
- BUILD-1, PADL-1

Research Issues

- Models & representations
- Fundamental algorithms
- Applications
- System architectures
- (User interfaces)
- (Characterize the problem domain!)

The Second Generation of Systems

- COMPAC, PROREN, EUCLID
- BUILD-2, ROMULUS
- GMSOLID, PADL-2

How are **solids** to be **represented** with symbols (e.g. in computers)?

What is a representation?
Properties of representations?

What is a solid, in mathematical terms?

PROPERTIES TO BE CAPTURED MATHEMATICALLY

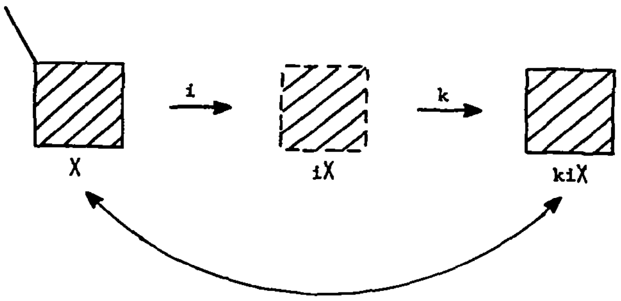
- RIGIDITY
- SOLIDITY
- FINITENESS
- CLOSURE UNDER BOOLEAN OPERATIONS
- FINITE DESCRIBABILITY
- BOUNDARY DETERMINISM

REGULARITY

X IS REGULAR IFF

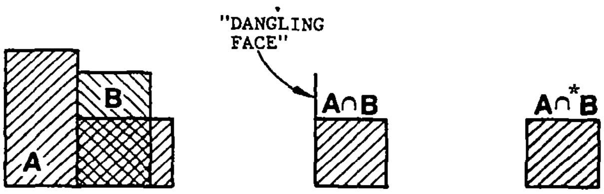
$X = kiX$

CLOSURE   INTERIOR



≠, HENCE X NOT REGULAR

REGULARIZED SET OPS



DEFINITIONS: $X \cap^* Y = ki(X \cap Y)$

$X \cup^* Y = ki(X \cup Y)$

$X -^* Y = ki(X - Y)$

$c^* X = kicX$

PROPERTY: REGULAR SETS & REGULARIZED SET OPS ARE A BOOLEAN ALGEBRA

SEMI-ALGEBRAIC SETS

$S = S_1 \text{ OP } S_2 \text{ OP } S_3 \quad . \ . \ .$

$\text{OP} = \ \cap \,|\, \cup \,|\, - \,|\, c$

$S_i = \{ (x,y,z) \mid F_i (x,y,z) \leq 0 \}$

POLYNOMIAL

SEMI-ANALYTIC SETS

DEFINED SIMILARLY, BUT F_i ANALYTIC

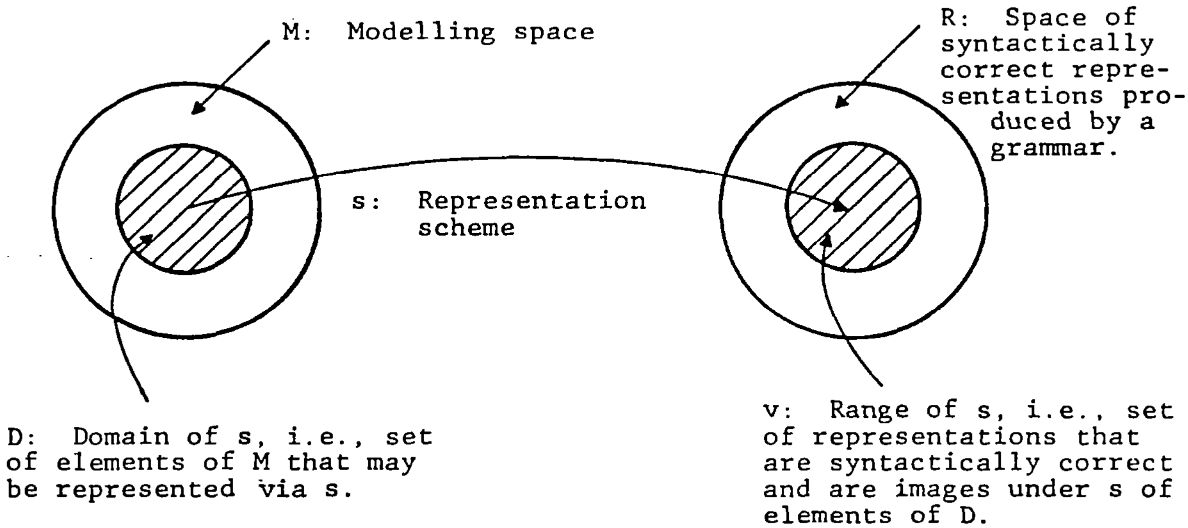
R-SETS

SUBSETS OF E^3 THAT ARE

- BOUNDED
- REGULAR (HOMOGENEOUSLY 3-D)
- SEMI-ANALYTIC

INTUITIVELY:

R-SETS ARE "CURVED POLYHEDRA" WITH "WELL-BEHAVED" BOUNDARIES



FORMAL PROPERTIES

- DOMAIN: WHAT CAN BE REPRESENTED?
- VALIDITY: DOES A REP. CORRESPOND TO SOME (AT LEAST ONE) SOLID?
- COMPLETENESS: DOES A REP. CORRESPOND TO A SINGLE SOLID?
- UNIQUENESS: DOES A SOLID HAVE A SINGLE REP.?

FUNDAMENTAL ALGORITHMS.

Set Membership Classification

$$M(X,R) = \{X\text{in}R, X\text{on}R, X\text{out}R\}$$

Numerical Geometry

Surf ∩ Surf

Curve ∩ Surf

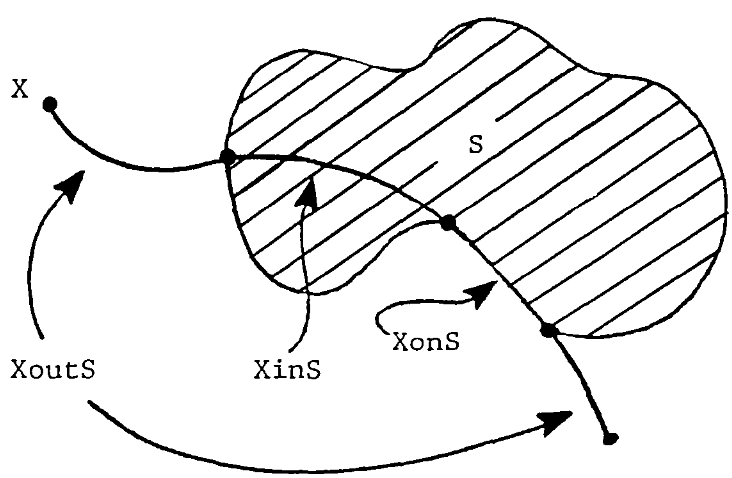
. . .

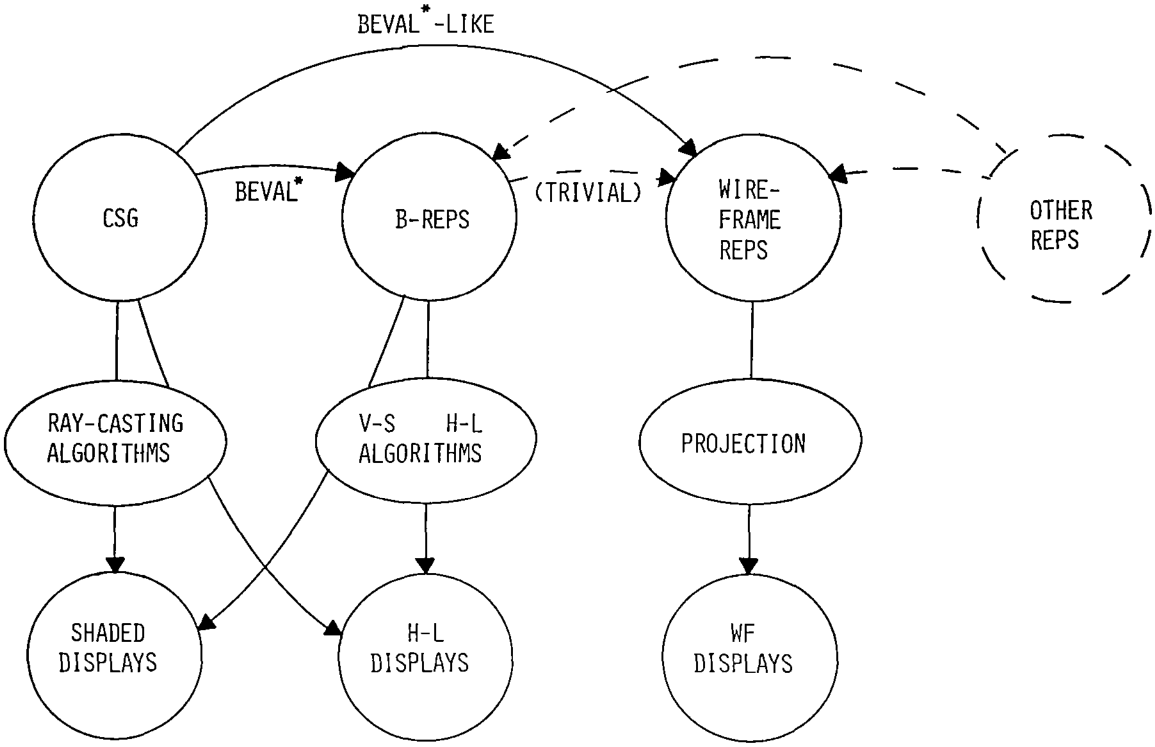
Boundary Evaluation & Composition

Polyhedral Approximation; Subdivision

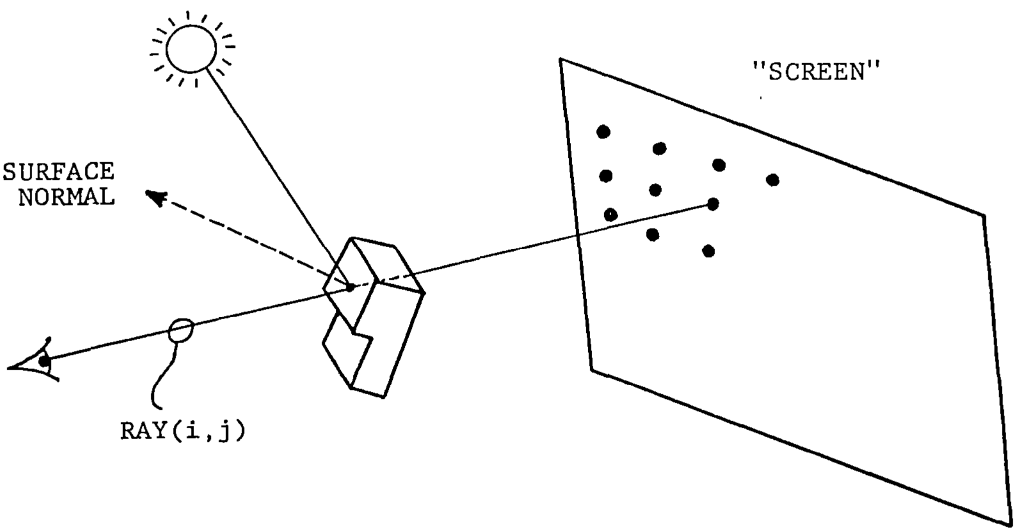
Topological Operations

M [X,S] =' (XinS, XonS, XoutS)

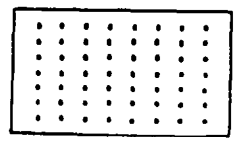




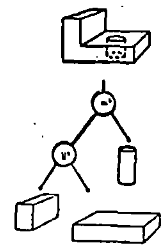
*"BEVAL" MEANS BOUNDARY EVALUATION, WHICH CONNOTES CONVERSION OF CSG REPRESENTATIONS INTO BOUNDARY REPRESENTATIONS.



EFFICIENCY OF RAY-CASTING SHADERS



$n_1 * n_2$ Pixels



N Primitive Instances

Naive Algorithm: TIME $\propto (n_1 * n_2) * N^2$

EVALUATION OF INTEGRAL ("MASS") PROPERTIES OF SOLIDS

$$\text{INTEGRAL PROPERTY} \stackrel{\Delta}{=} \int_S f(p)\, dV$$

A SOLID

DIFFERENTIAL VOLUME

VECTOR (POINT) IN E^3

POLYNOMIAL FUNCTION

EXAMPLES:

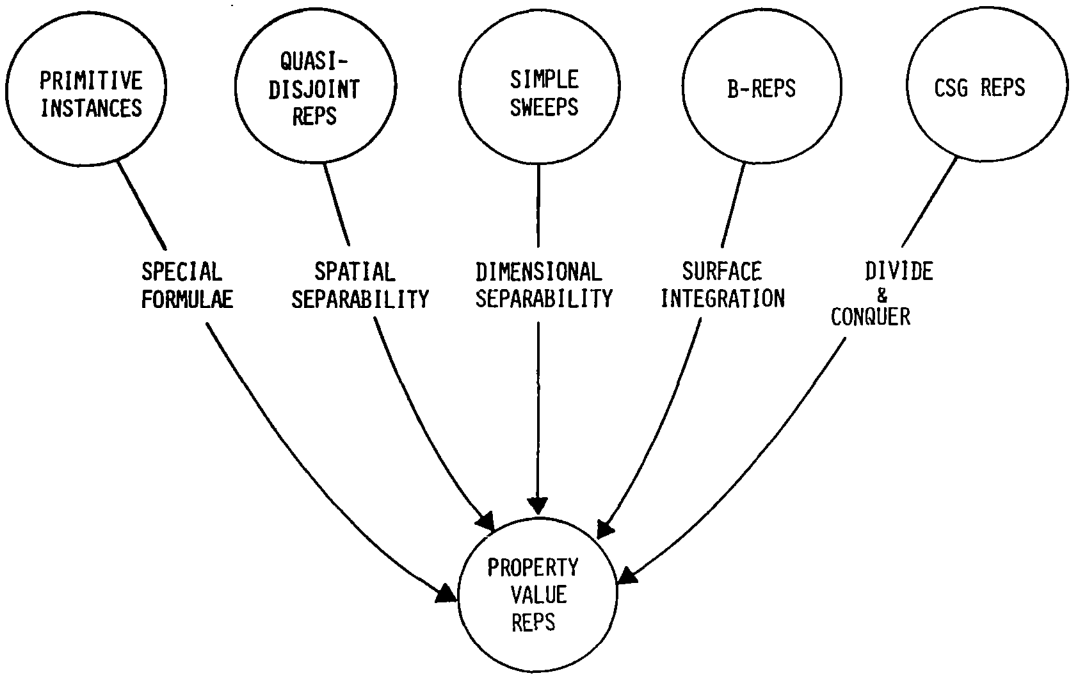
$$V = \int_S dV$$

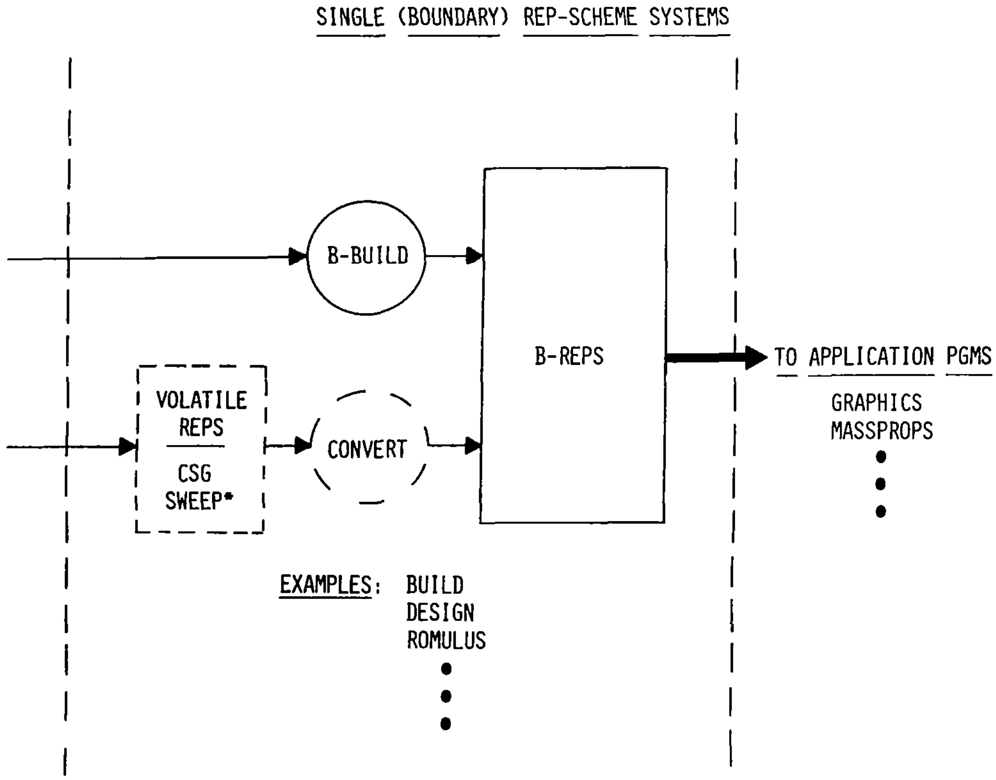
$$CG_x = \frac{1}{V} \int_S x\, dV$$

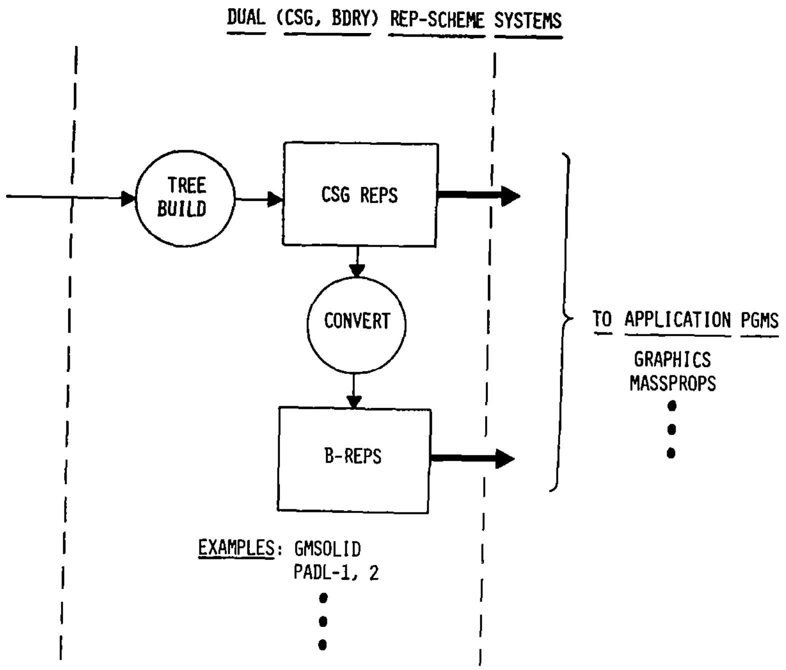
$$I_{xx} = \rho \int_S (y^2 + z^2)\, dV$$

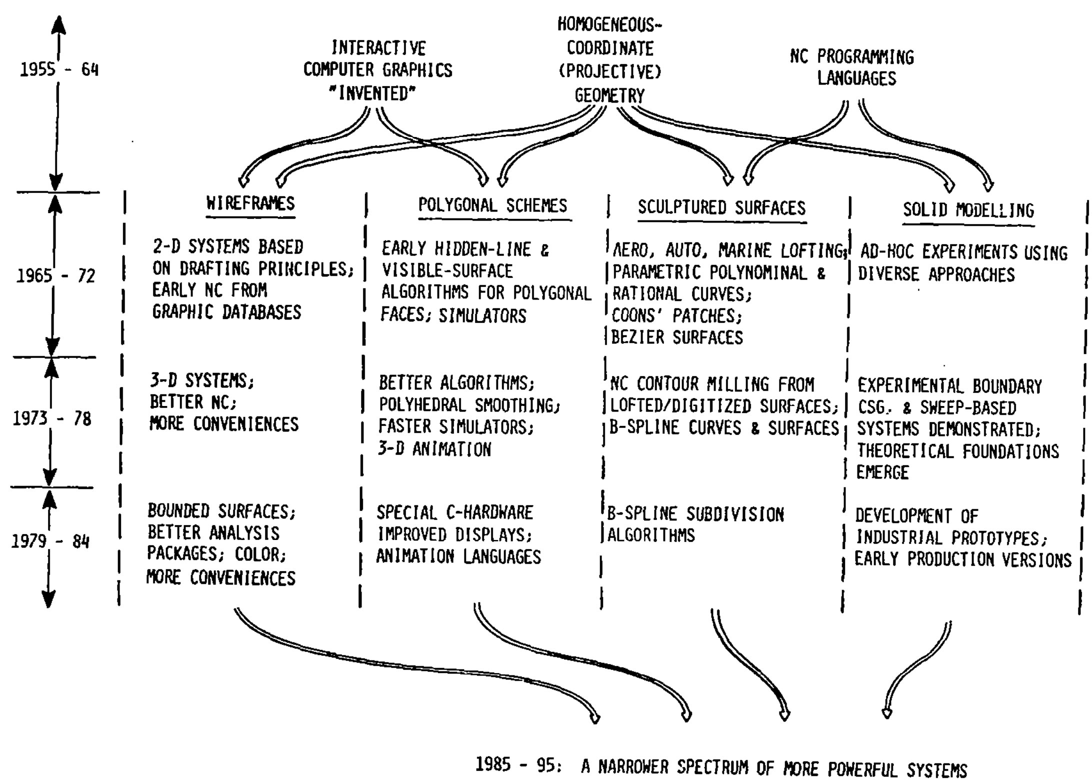
$$P_{xy} = \rho \int_S (xy)\, dV$$

$$\begin{bmatrix} I_{xx} & -P_{xy} & -P_{xz} \\ -P_{yx} & I_{yy} & -P_{yz} \\ -P_{zx} & -P_{zy} & I_{zz} \end{bmatrix}$$


PRIMITIVE INSTANCES
QUASI-DISJOINT REPS
SIMPLE SWEEPS
B-REPS
CSG REPS
SPECIAL FORMULAE
SPATIAL SEPARABILITY
DIMENSIONAL SEPARABILITY
SURFACE INTEGRATION
DIVIDE & CONQUER
PROPERTY VALUE REPS


SINGLE (BOUNDARY) REP-SCHEME SYSTEMS
B-BUILD
B-REPS
TO APPLICATION PGMS
GRAPHICS
MASSPROPS
VOLATILE REPS
CSG
SWEEP*
CONVERT
EXAMPLES: BUILD
DESIGN
ROMULUS


DUAL (CSG, BDRY) REP-SCHEME SYSTEMS
TREE BUILD
CSG REPS
CONVERT
B-REPS
TO APPLICATION PGMS
GRAPHICS
MASSPROPS
EXAMPLES: GMSOLID
PADL-1, 2


1955 - 64
INTERACTIVE COMPUTER GRAPHICS "INVENTED"
HOMOGENEOUS-COORDINATE (PROJECTIVE) GEOMETRY
NC PROGRAMMING LANGUAGES
WIREFRAMES
POLYGONAL SCHEMES
SCULPTURED SURFACES
SOLID MODELLING
1965 - 72
2-D SYSTEMS BASED ON DRAFTING PRINCIPLES; EARLY NC FROM GRAPHIC DATABASES
EARLY HIDDEN-LINE & VISIBLE-SURFACE ALGORITHMS FOR POLYGONAL FACES; SIMULATORS
AERO, AUTO, MARINE LOFTING; PARAMETRIC POLYNOMINAL & RATIONAL CURVES; COONS' PATCHES; BEZIER SURFACES
AD-HOC EXPERIMENTS USING DIVERSE APPROACHES
1973 - 78
3-D SYSTEMS; BETTER NC; MORE CONVENIENCES
BETTER ALGORITHMS; POLYHEDRAL SMOOTHING; FASTER SIMULATORS; 3-D ANIMATION
NC CONTOUR MILLING FROM LOFTED/DIGITIZED SURFACES; B-SPLINE CURVES & SURFACES
EXPERIMENTAL BOUNDARY CSG, & SWEEP-BASED SYSTEMS DEMONSTRATED; THEORETICAL FOUNDATIONS EMERGE
1979 - 84
BOUNDED SURFACES; BETTER ANALYSIS PACKAGES; COLOR; MORE CONVENIENCES
SPECIAL C-HARDWARE IMPROVED DISPLAYS; ANIMATION LANGUAGES
B-SPLINE SUBDIVISION ALGORITHMS
DEVELOPMENT OF INDUSTRIAL PROTOTYPES; EARLY PRODUCTION VERSIONS
1985 - 95: A NARROWER SPECTRUM OF MORE POWERFUL SYSTEMS

Solid Modellers Available in the USA February 1983 With apologies for inadvertent omissions		
MODELLER	**VENDOR**	**CORE SOFTWARE**
BAUSTEIN GEOMETRIE	Sperry Univac	COMPAC (T.U. Berlin)
CATIA	IBM	Dassault (France)
CDC – Synthavision	CDC	Synthavision (MAGI)
DDM – Solids	Calma	
EUCLID	Matra Datavision DEC	CNRS (France)
GEOMOD	SDRC General Electric	SDRC
MEDUSA	Prime (Computervision?)	CIS (U.K.)
PADL-2	U. Rochester	
PATRAN-G	PDA Engineering	
ROMULUS	Evans & Sutherland	ShapeData (U.K.)
SOLIDESIGN	Computervision	
SOLIDS MODELING	Applicon	Synthavision (MAGI)
SYNTHAVISION	MAGI	
TIPS-1	CAM-I	Hokkaido U.
UNISOLIDS	McAuto	PADL-2 (U. Rochester)

SOLID MODELLING EXTENSIONS

Domain Extensions

- Sculptured surfaces
- Blending

Variational Geometry

- Dimensions & tolerances
- Geometric constraints

Improving System Performance

- Better algorithms
- Exploitation of locality
- Special hardware

User Interfaces

- "Poking"
- "Programming"

Fundamental Issues & Algorithms

- Approximation strategies
- New representation conversions
- New generic-problem algorithms, e.g. NOD, SOD
- "ϵ-problems" in numerical geometry

APPLICATIONS IN DESIGN

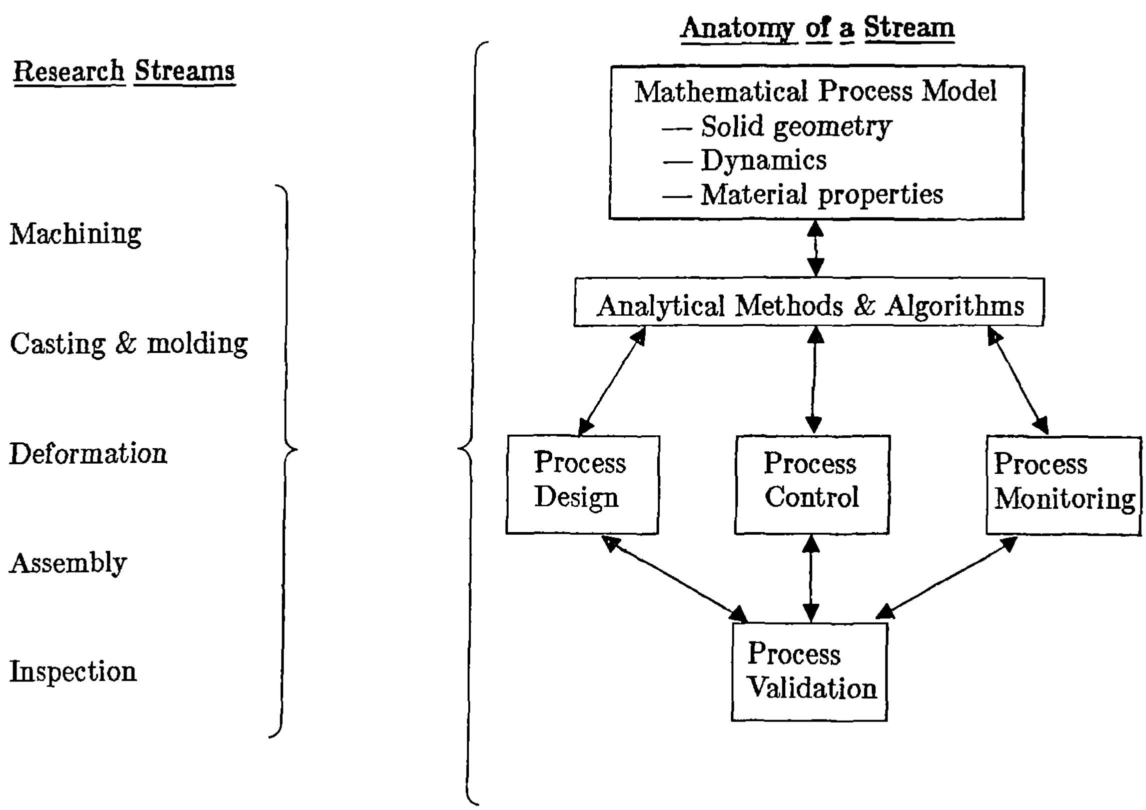
Kinematics & Dynamics

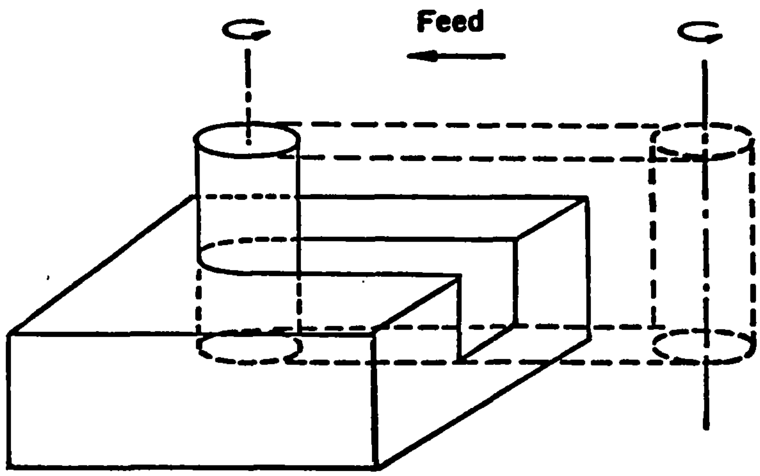
- Interfaces to IMP, ADAMS, DRAM, . . .
- Interference analysis

Finite Element Analysis

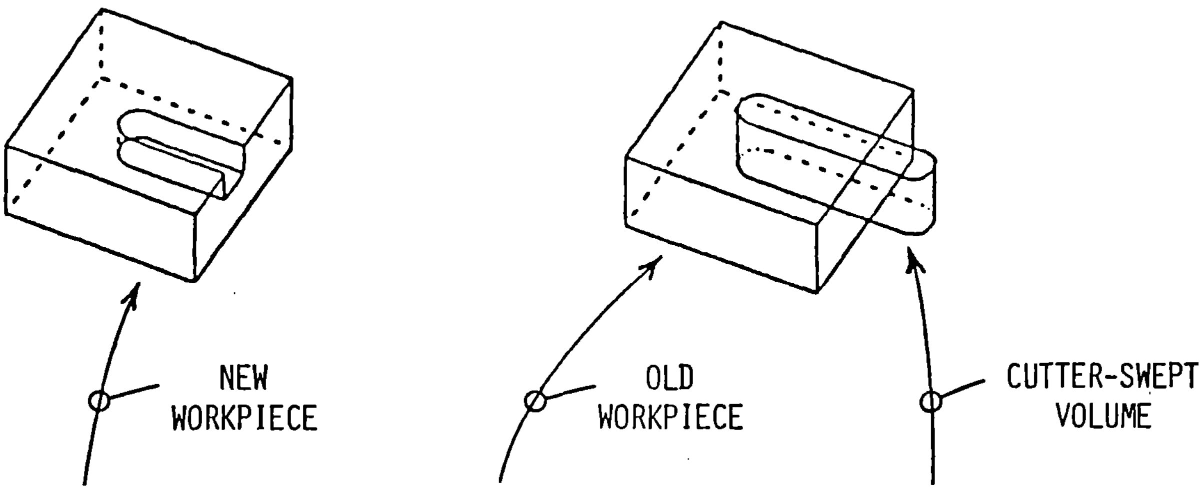
- The auto-meshing problem
- (A new look at analytical methods)

APPLICATIONS IN PRODUCTION

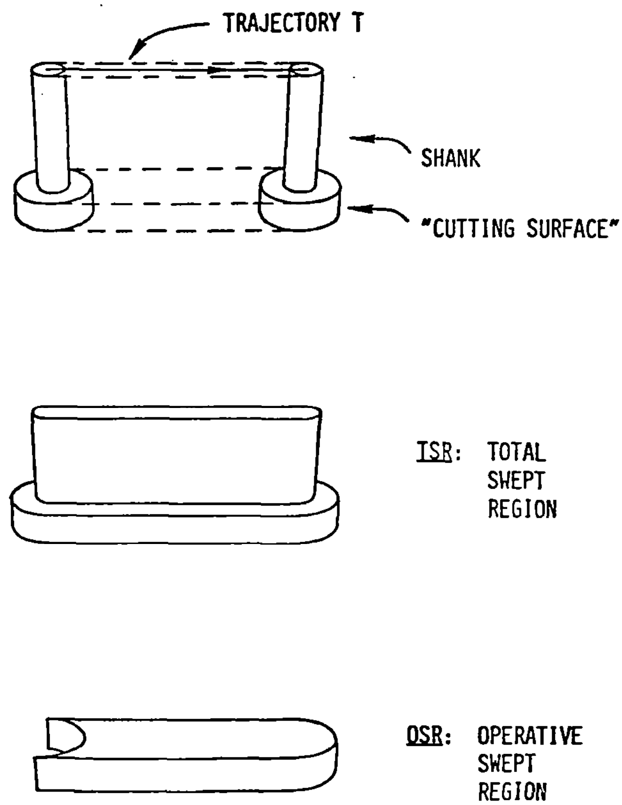




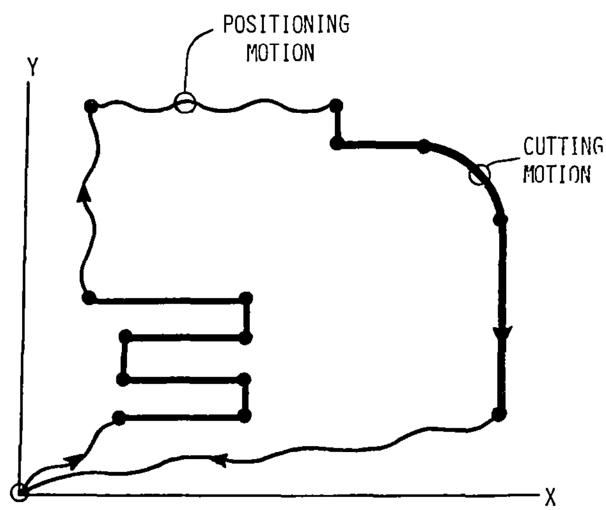
SINGLE AXIS MILLING

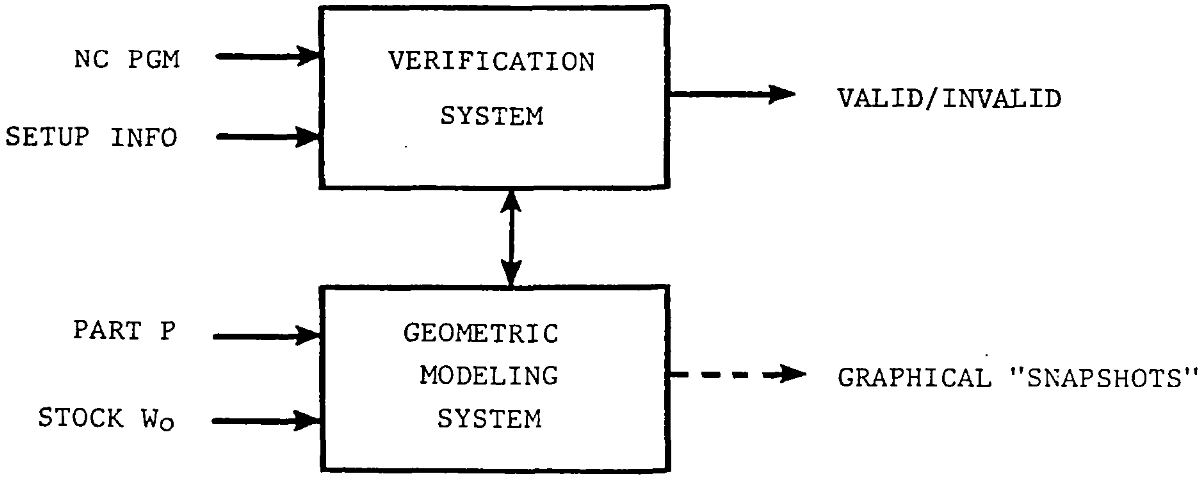


$$W_I \quad = \quad W_{I-1} \quad .DIF. \quad V_I$$



REGIONS SWEPT BY AN END MILL





VERIFICATION BY SIMULATION -- A FIRST-ORDER ALGORITHM

```
WORKPIECE ← STOCK
DO UNTIL (END OF PROGRAM OR ERROR)
     READ NC COMMAND
     IF (MOTIONAL COMMAND) THEN
          IF (PRECONDITIONS) THEN       }
              UPDATE WORKPIECE          }  PROCESS MODEL
              UPDATE CUTTER POSITION    }
          ELSE ERROR
          END-IF
     END-DO
IF (WORKPIECE ≠ SPECIFIED PART) THEN ERROR
```

Validity Conditions

GVC1/PC: $TSR_i \cap^* J = \emptyset$ (Fixture Collision)

GVC2/P : $TSR_i \cap^* W_{i-1} = \emptyset$ (Workpiece Collision)

GVC3/C : $V(t_o, i) \cap^* W_{i-1} = \emptyset$ (Cutter Position)

GVC4/C : $TSR_i \cap^* (W_{i-1} -^* OSR_i) = \emptyset$ (Cutter Clearance)

TVC1/C : Admissible-Feed-Direction

TVC2/C : Admissible-Feed-Rate

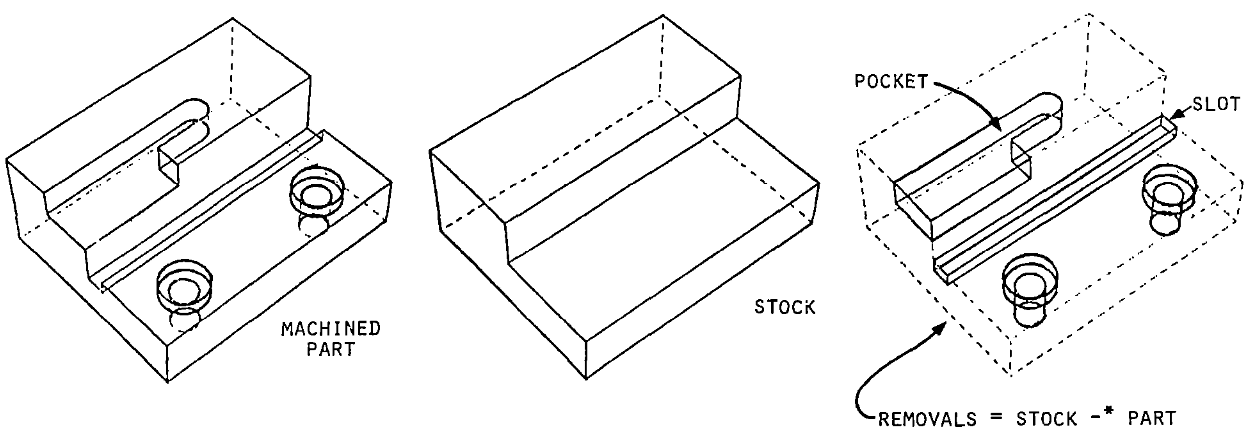
TVC3/C : Tolspecs-Statisfied

Correctness Conditions

CC1/C : $TSR_i \cap^* P = \emptyset$ (Invasive Machining)

Effectiveness Condition

EC1/C : $OSR_i \cap^* W_{i-1} \neq \emptyset$



EDUCATION

A Basic Distinction

— User education

— vs.

— Implementor education

Professional Updating

— Users → Short Courses

— Implementors → Residencies

Engineering Degree Programs

— User education

Integrate the new tools into established courses, don't segregate them in "CAD/CAM courses"

— Implementor education

Specialized & cross-disciplinary; not all schools need have it

SOLID MODELLING IN THE DEPARTMENT OF ENERGY

D. P. Peterson
Sandia National Laboratories
Albuquerque, New Mexico

DEPARTMENT OF ENERGY/NUCLEAR WEAPONS COMPLEX ELEMENTS

The Department of Energy/Nuclear Weapons Complex (DOE/NWC) is comprised of three R&D Laboratories (Los Alamos National Laboratory (LANL), Lawrence Livermore National Laboratory (LLNL) and Sandia National Laboratories (SNL) and several production agencies: e.g., Bendix Corporation (Kansas City Plant) and Union Carbide Corporation (Y-12 Plant). Design definition information must be transmitted among the R&D laboratories and the production agencies. The growing importance of CAD/CAM has highlighted both the advantages and the problems of exchanging this information.

SANDIA NATIONAL LABORATORIES

Early in 1982, we undertook to evaluate three solid modellers (SMs) for our use. The three SMs that were selected are the Part and Assembly Description Language (PADL-2)*, developed by the Production Automation Project at the University of Rochester (ref. 1), the Technical Information Processing System (TIPS), developed at Hokkaido University (ref. 2), and a version of Synthavision, developed by the Mathematical Applications Group, Inc. (ref. 3), which is interfaced to the Applicon Graphic System and is hereafter denoted as AGS/S. Each of these SMs uses a constructive solid geometry representation (csg-rep) as its primary representation; however, there are significant differences among them.

We will discuss the nature of these SMs and the reasons for their selection, and convey our findings and opinions about their merits and deficiencies.

SNL uses existing wireframe CAD systems extensively and these systems will continue to play an important role in our design activities. Thus, it is important to smoothly interface our existing systems with the more powerful Geometric Modelling Systems (GMS) that use an SM. One way to attain this interface is to use the GMS to generate a wireframe definition for the existing CAD systems. This permits us to have the benefits of the SM definition of a part and still easily introduce its definition into the existing CAD/CAM structure. One example of this benefit is that we can then use an

---

* PADL-2 was developed with joint funding from the National Science Foundation and a consortium of 10 industrial firms: Eastman Kodak Co., Digital Equipment Corp., Calma Co., Boeing Commercial Airplane Co., McDonnell Douglas Automation Co., Sandia National Laboratories, Deere & Co., United Technologies Corp., Tektronix, Inc., and Lawrence Livermore National Laboratory.

existing interface into ANVIL 4000 to help generate a cutter location (CL) file. Experience with getting this information from AGS/S and PADL-2 will be discussed. (TIPS does not yet provide this capability.)

SNL has a need for presenting graphic displays of new designs, and the generation of these displays by conventional graphic art techniques is very time consuming. Using displays of the AGS wireframe has been found helpful. Even better, the AGS wireframe has been used as a starting point to generate input to MOVIE.BYU, which permits colored displays of a wireframe-defined part to be readily obtained. (Most SMs can now provide this kind of output, but MOVIE.BYU provides some distinct advantages with respect to computer time.) This activity, as well as some preliminary work on developing a computer algorithm to transform directly a PADL-2 boundary representation (b-rep) into input for MOVIE.BYU, will be presented. It is not clear how feasible or attractive this latter approach will be but it has provided some interesting lessons in mapping between different representations.

LAWRENCE LIVERMORE NATIONAL LABORATORIES

LLNL has different needs for an SM than does SNL.

LLNL has found that TIPS has provided them with a useful capability to model some of their parts. They have successfully used its mass properties capability to demonstrate how an SM can help in the design of a large part. A shaded graphic display capability has been developed for TIPS which provides good visual output. This is an area in which TIPS has been deficient.

LLNL is planning to continue its efforts to utilize TIPS to help solve their problems.

BENDIX CORPORATION

Each R&D lab has its own needs, but a production agency, such as Bendix, must be able to accommodate the definitions from each of the R&D labs.

Bendix has obtained a version of Synthavision that has been modified by Control Data Corporation (CDC/S). This version differs significantly from AGS/S. For example, the input mechanism for this implementation is by a procedural language rather than by graphic input. Bendix has been successful with simple parts in using the edge file from CDC/S as input to CD-2000, with subsequent generation of a cutter location file.

Bendix personnel observed that there exists a large (and increasing) inventory of computer files of parts that have been defined on wireframe CAD systems; these files will exist and be accessed for many years. It would be desirable to automatically transform these definitions into SM definitions so that the full power of the GMS can be used without having to redefine the part. Work done to generate a set of consistent b-reps from the wireframe definition of a part will be mentioned.

OTHER DOE/NWC FACILITIES

The needs and activities of some of the other organizations will be briefly mentioned.

REFERENCES

1. Brown, C. M.: PADL-2: A Technical Summary. IEEE Computer Graphics and Applications, Vol. 2, pp. 69-84, March 1982.

2. Wang, K. K., ed.: TIPS-1 Geometric Modeling Documentation and Software. Computer Aided Manufacturing International, Inc., Report R-81-GM-04, Vols. 1-4, 1981.

3. Goldstein, R.: Defining the Bounding Edges of a Synthavision Solid Model. Proc. 18th Design Automation Conf., IEEE, pp. 457-461, June 1981.

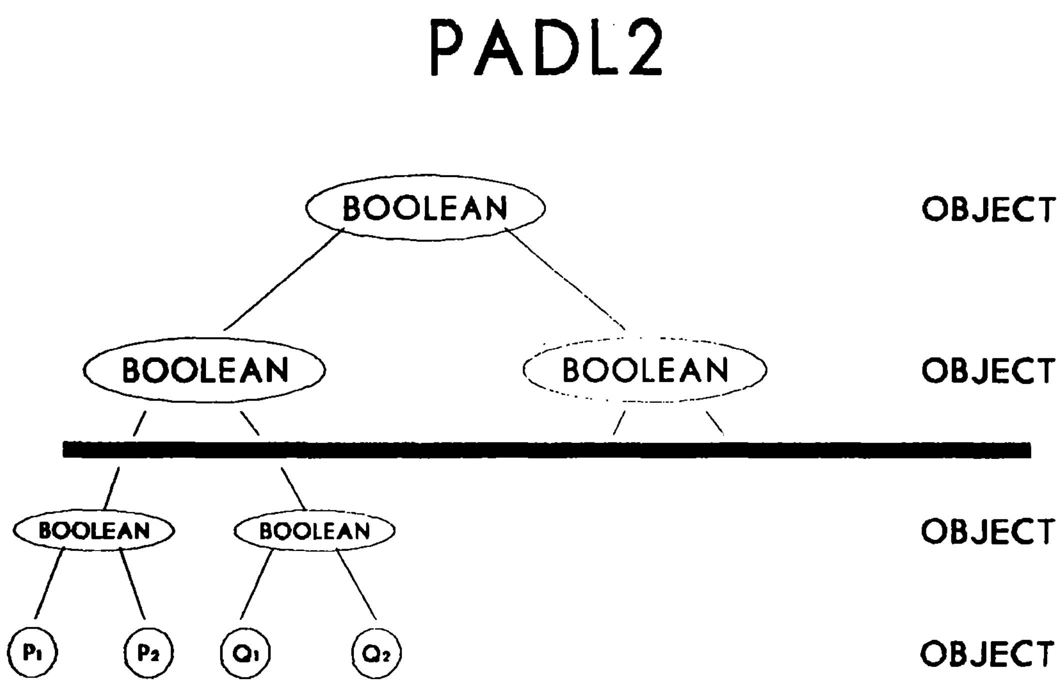


Figure 1

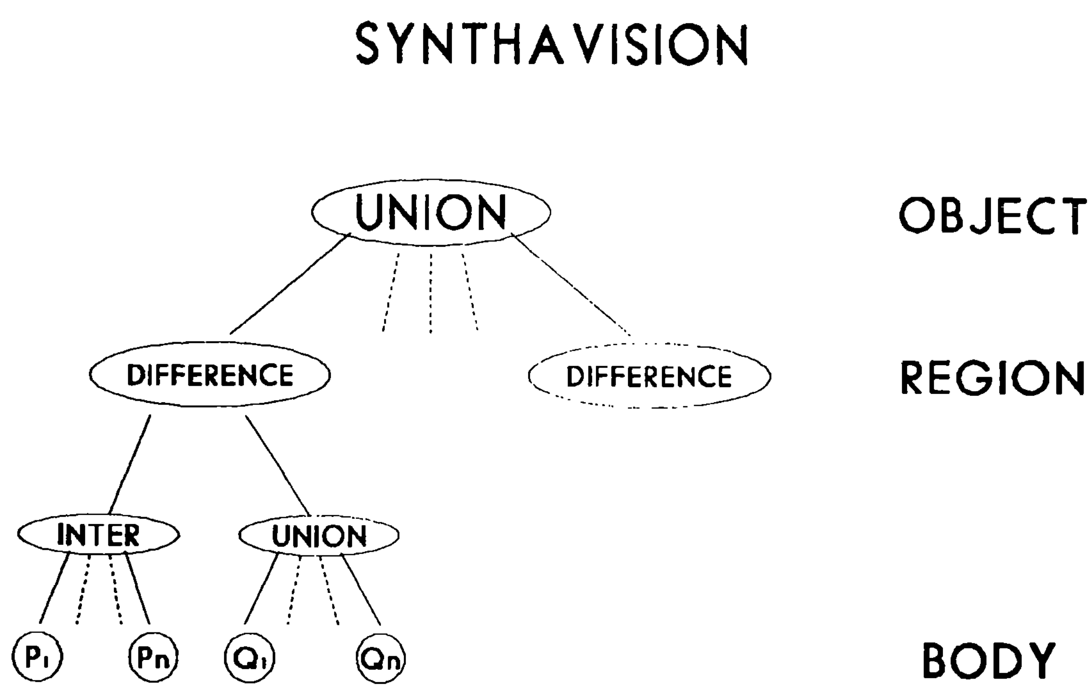


Figure 2

TIPS

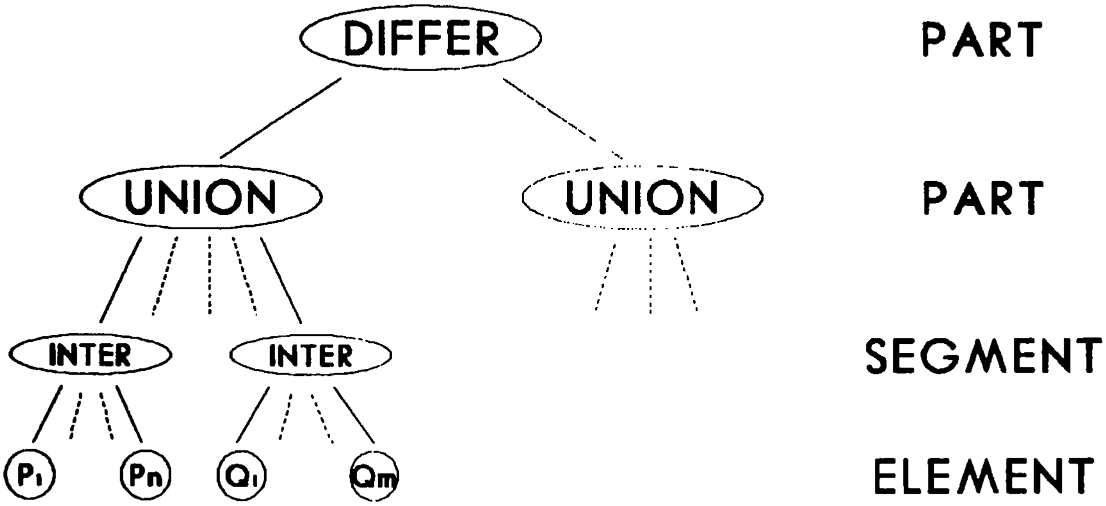


Figure 3

IDENTIFY AND COMPARE CURRENT CAPABILITIES

ALL MODELLERS HAVE THE FOLLOWING:

Geometric Coverage: Plane, Cylinder, Sphere, Cone
Accuracy & Correctness: Accurate & "Correct" Input
Application Programs: Mass Properties, Hidden Line

SOLID MODELLER	Geometric Coverage	Accuracy & Correctness	Application Programs
PADL2	Adequate	Good	Wireframe, Color Display
SYNTHAVISION	Good	Unreliable	Wireframe, Color Display
TIPS	Excellent	Unreliable	N/C, 2-D FEM

Figure 4

POTENTIAL OF THE SOLID MODELLERS

SOLID MODELLER	Graphic Interface (Interactive)	Editing Definitions	Component Library	Handling Assemblies
PADL2	Good	Good	Excellent	Excellent
SYNTHAVISION	Poor	Fair	Fair	Good
TIPS	Poor	Poor	Poor	Good

Figure 5

VOLUMETRIC REPRESENTATION FOR OBJECT MODEL ACQUISITION[1]

W. N. Martin
University of Virginia
Charlottesville, Virginia

B. Gil and J. K. Aggarwal
University of Texas
Austin, Texas

I.

This paper describes a volumetric representation that has been specified for use in a system that derives object models from a sequence of images with viewpoint specifications. The volumetric representation is an enumerative structure (ref. 1, 2, 3) referred to here as a "volume segment" representation. The model acquisition system was designed to form an initial object approximation, then refine that approximation as subsequent information is obtained (see also ref. 4). The refinement process requires a readily modifiable object representation, while the overall goal of the system demands a representation descriptive of surface detail. The "volume segment" structure provides both of those capabilities.

The remainder of this paper will define the volume segment representation, briefly describe the model acquisition system, and then present an example as analyzed by the system.

II.

The volume segment representation is based on volume primitives that are rectilinear parallelepipeds with uniform width and depth, but varying height. The height dimensions of all the primitives are parallel to the y-axis of the model coordinate system, while the width and depth dimensions are parallel to the model x- and z-axes, respectively. With two of the dimensions specified as being uniform, each volume primitive can be represented by a single line segment, parallel to the y-axis, the extent of which is the height of the primitive.

The volume primitives are grouped by x-coordinate into collinear line segments. The x-coordinate sets are then grouped by z-coordinate into sets of coplanar lines. Those sets are maintained as three levels of ordered linked lists. The first level is a list of z values. Attached to each z-element is a linked list of x values, and for each x-element there is a list of line segments. Each segment is specified by a pair of values which are the y-coordinates of the endpoints of the line segment. In this way a complex volume

[1] This research was supported in part by AFOSR grant 82-0064.

can be modelled through volume primitives represented by a set of parallel line segments in a hierarchically ordered list structure.

III.

The model acquisition process operates on a sequence of images. In particular, each element of the sequence is a binary image from which the occluding contour of the object can be extracted, along with a specification of the viewpoint orientation for that image. An orthogonal projection along the y-axis of the viewpoint coordinate system is assumed throughout this discussion. To see how a volume segment model can be derived from that sort of data, consider Figure 1. Recall that the first two elements of the sequence are used to create the initial representation which will be an approximation of that refined by the remaining elements.

Shown in Figure 1 are two occluding contours with corresponding coordinate axes indicating the viewpoint orientations. Also shown are projection rays (parallel to the appropriate viewline) passing through the occluding contours. Clearly, for a single contour the locus of these projection lines forms the surface of a volume that necessarily contains the actual object. The problem is that the volume is infinite in extent. However, if a second contour is given from a distinct view (non-parallel viewline) a second volume will be specified. The intersection of the two volumes will be a finite volume bounding the original object.

The initial model is created from the bounding volume of the first two contours. The axis system of the model is defined with an arbitrary point as origin, the y-direction parallel to the viewline of the first image, the z-direction parallel to the cross-product of the viewlines from the two images, and the x-direction completing a right-handed system. Within this coordinate system the volume primitives that fill out the bounding volume can be readily specified, the endpoints of the corresponding line segments computed, and the hierarchical structure created.

The remaining images of the sequence refine the initial model in a manner indicated by Figure 2. Again, each element of the sequence provides an occluding contour and a viewpoint specification. Figure 2 contains an example volume segment representation in the model coordinate system, a new contour in the image axis system, and the projection (along the new viewline) of the model y-axis unit vector into the new image plane. Also shown are "raster lines" (ref. 5) of the contour area with a raster direction parallel to the projected unit vector. To refine the model, given a new image, the system projects each volume segment into the new image plane, clips the projected segment to the rasterized contour and then reprojects the clipped line in place of the original segment. "Rasterizing" the contour area simplifies the clipping process. It is important to note that while the refinement process modifies the hierarchial structure, the volume primitives remain homogeneously rectilinear parallelepipeds.

IV.

Figures 3 through 7 display an example as processed by the system. In this example two objects are in the scene: a block T behind a block O. Figures 3 through 5 are the input image sequence elements. Figure 6 is the initial volume segment representation, while Figure 7 is a refined model. In these figures surface models are shown because the resulting diagrams are easier to interpret visually. The last figure shows that extraneous elements of the initial model are being removed by the refinement process.

The transformation from the volume segment representation which does not explicitly contain complete adjacency information to a surface representation highlights the important concept that no single representation will be suitable for both the acquisition and manipulation processes in object modelling. The representation must be appropriate to the particular task to be performed, with effective procedures for transforming one representation to another.

REFERENCES

1. A. A. G. Requicha, "Representations for rigid solids: Theory, method and systems," Computing Surveys, vol. 12, no. 4, 1980, pp. 437-464.

2. J. K. Aggarwal, L. S. David, W. N. Martin and J. W. Roach, "Survey: Representation methods for three-dimensional objects," in <u>Progress in Pattern Recognition</u>, vol. 1, L. Kanal and A. Rosenfeld, Editors, North-Holland Pub. Co., Amsterdam, The Netherlands, 1981, pp. 337-391.

3. C. L. Jackins and S. L. Tanimoto, "Oct-trees and their use in representing three-dimensional objects," Computer Graphics and Image Processing, vol. 14, 1980, pp. 249-270.

4. B. G. Baumjart, "Geometric Modeling for Computer Vision," Stanford AI Lab Memo AIM-249, Stanford University, Oct. 1974.

5. W. E. Newman and R. F. Sproull, <u>Principles of Interactive Computer Graphics</u>, Second Edition, McGraw-Hill, New York, 1979, pp. 229-245.

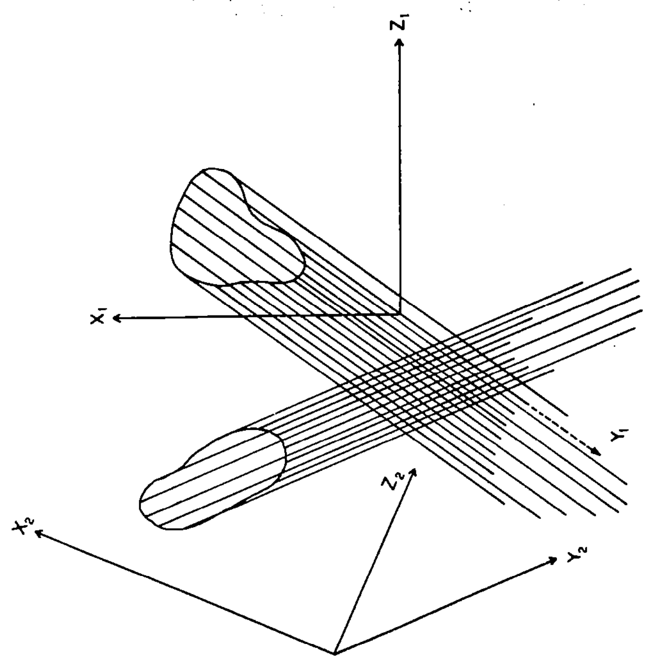


Figure 1.- Two views.

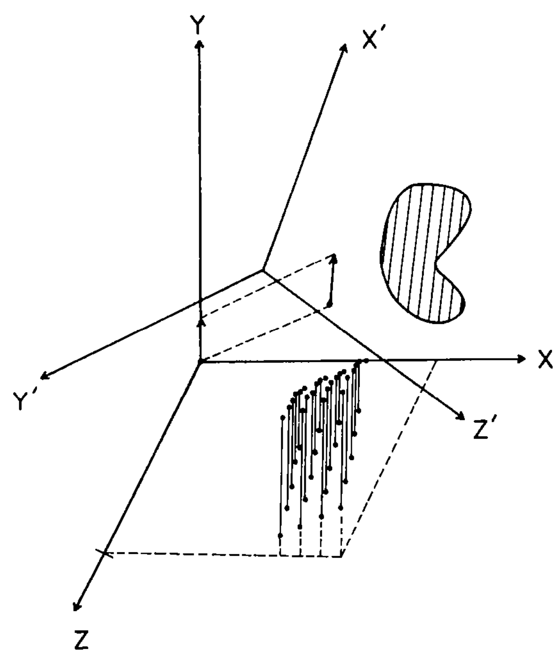


Figure 2.- Additional view for refinement.

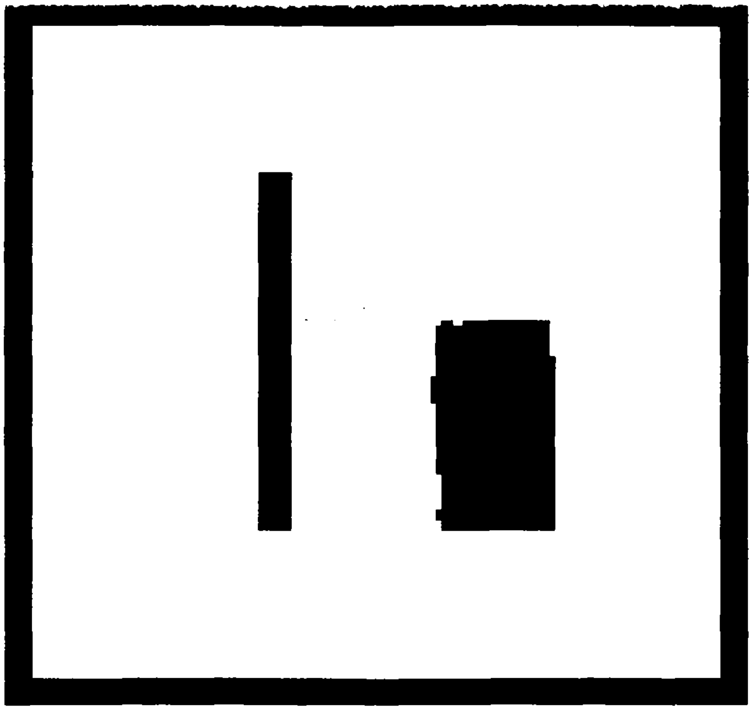


Figure 3.- Image from first view.

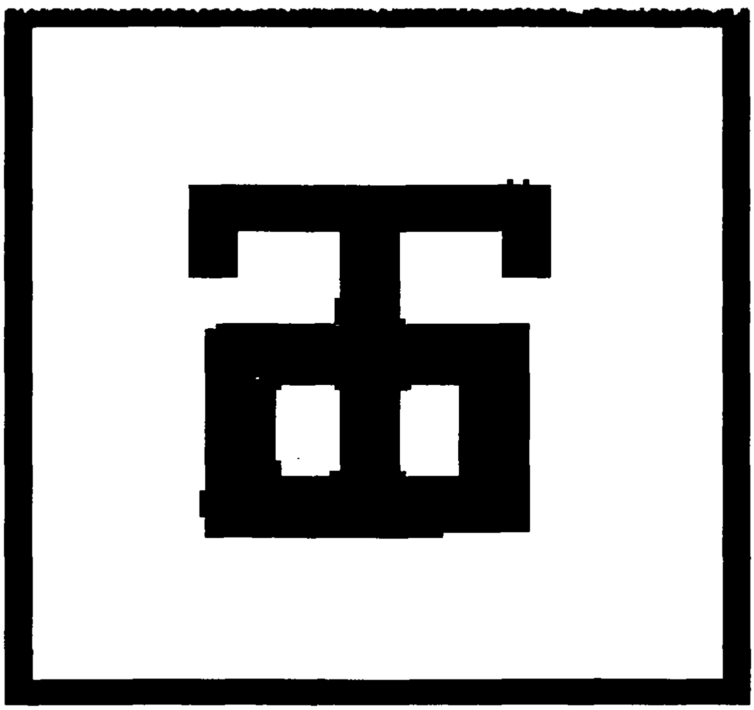


Figure 4.- Second image.

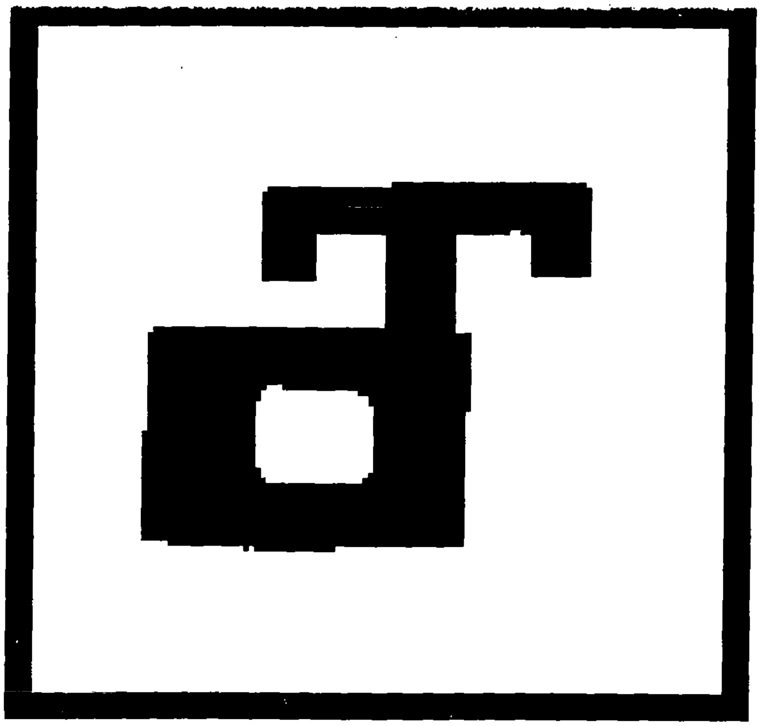


Figure 5.- Third image.

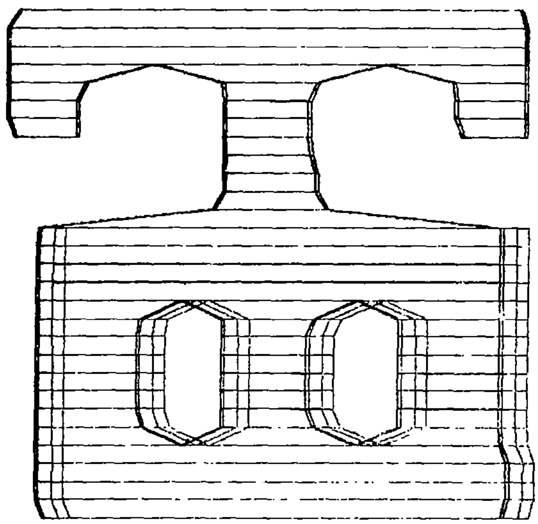


Figure 6.- Initial volume representation.

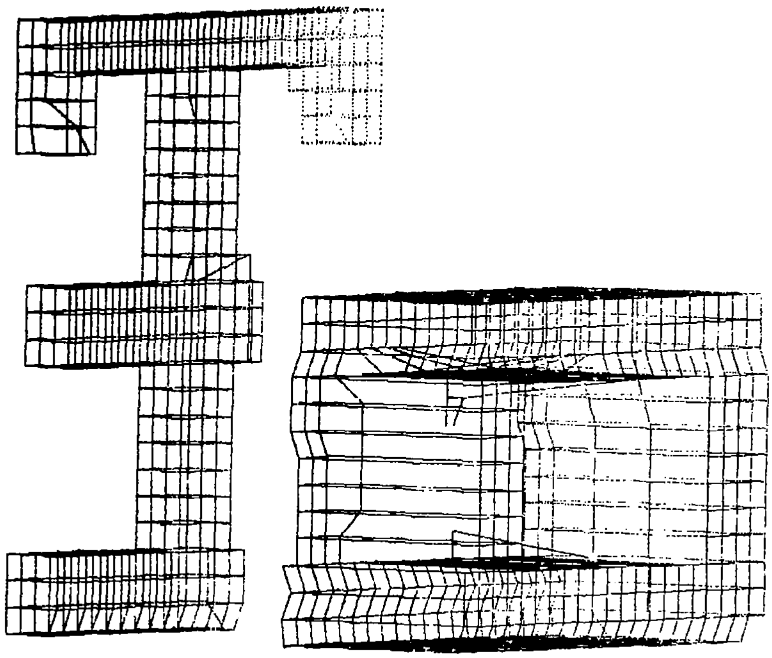


Figure 7.- Refined volume representation.

The Representation of Manipulable Solid Objects in a Relational Database *

Dennis Bahler
Department of Applied Mathematics
and
Computer Science
UNIVERSITY OF VIRGINIA
Charlottesville, VA

Introduction

This project is concerned with the interface between database management and solid geometric modeling. The desirability of integrating computer-aided design, manufacture, testing, and management into a coherent system is by now well recognized. One proposed configuration for such a system uses a relational database management system as the central focus; the various other functions are linked through their use of a common data representation in the data manager, rather than communicating pairwise [1, 2]. Our goal is to integrate a geometric modeling capability with a generic relational data management system in such a way that well-formed questions can be posed and answered about the performance of the system as a whole. One necessary feature of any such system is simplification for purposes of analysis; this and system performance considerations meant that a paramount goal therefore was that of unity and simplicity of the data structures used. One seeks a standard unified way of representing geometries such that the geometry may be used in a variety of engineering applications in addition to simple display.

Many existing geometric modeling systems [3, 4, 5, 6, 7] as well as the IGES standard [8] define a large number of different geometric entities, each with a different representation. One then has a variety of procedures and relationships whose applicability is dependent on the particular entities involved. The approach of this study is different. We have defined a single canonical representation for <u>all</u> objects, and are attempting to establish to what extent this representation enables adequate and efficient solution of geometry-related engineering problems.

Specifically, for purposes of analysis we have defined a class of 3-dimensional solid objects in terms of their data representation, and are exploring the implications of that representation with respect to a class of manipulative operations that we wish to perform on such objects. All objects in the system consist of subsets of 3-space enclosed by surfaces defined in terms of

uniform periodic cubic B-spline curves, and
interpolation between pairs of such (closed) curves.

In essence, all solid objects are approximated by "cylinders", whose end surfaces can be delimited by closed B-spline curves. The B-spline curves are represented in the database as the set of vertices of their controlling polygons. The cardinality of this set is fixed as a system parameter.

This approach carries a number of interesting implications. For one, a single--and quite simple--database schema can be used to represent a relatively large class of objects. The model is very storage-efficient, while being flexible

* This work was supported by NASA under contract NAG 5-28209.

enough to enable a large class of verification and manipulation operations. Secondary storage accesses, particularly if under the control of a relational database manager, often constitute a system bottleneck which is orders of magnitude larger than computation times. The use of B-splines as the sole representational technique greatly simplifies the data management/data structure aspects of the system. It does so at the cost of some additional computational complexity in those routines which interface the geometric modeler with the data management system. One of the objects of this study is to explore whether storage accesses can be significantly reduced without rendering these computation times unacceptably large. In a sense, we are seeking to transform what is potentially a seriously IO-bound problem into a problem in which secondary accesses are less significant.

A second centrally important feature of such a system is that it reduces to one the number of routines which the system must support to perform a given operation on objects. Instead of maintaining a different routine for each type of entity, it is now necessary to devise only a single algorithm, which can operate on all objects in the system. It should be understood that, for objects with simple geometry, such an algorithm is likely to be more complex than a special-purpose routine. Its attraction lies in its universal applicability within the context of this system.

The project is investigating the feasibility under the model of such operations as:

(1) Detection of the congruence of two complete, closed curves. This may be done by testing the points of their associated control polygons for coincidence to within some epsilon when one is mapped on the other. One question which arises is whether it is cheaper in terms of overall performance to store congruence information in the database or to calculate it whenever needed.

(2) Detection of the congruence of two objects. Object congruence depends upon the congruence of pairs of the curves involved, and upon the relative distance and spatial orientation of the curves.

(3) Detection of the position of a given point in space relative to an object.

(4) Decomposition of an object, and subsequent automatic reassembly by identification of previously adjacent surfaces. This operation makes use of the congruence detection described above.

(5) Combination of objects to construct more complex entities by means of a binary set union or "gluing" operator. Two objects may be joined along the interpolated dimension if their associated polygons exhibit partial congruence such that the curves have at least one congruent segment. They may be joined along the other dimension if the curves involved are coplanar, or if they have congruent associated control polygons.

(6) Intersection of an object with a plane. Linear interpolation may be used to determine the virtual face of an object produced when it is cut by a plane [9], and a linear system may be solved to produce the control polygon associated with that face.

(7) Calculation of area and volumetric properties of an object. These calculations are far more complex for objects as we have defined them than for most objects of "simple" geometry, but our system need not maintain a routine for each type of defined entity.

The use of splines

The use of cubic B-spline curves [10, 11] is central to this approach. In this section we briefly review their definition and properties and indicate our reasons for using them.

B-spline curves are piecewise polynomial curves defined by the vector equation

$$\underline{P}(u) = \sum_{i=0}^{n} \underline{p}_i \; N_{i,k}(u) \qquad (1)$$

where k is the order of the curve ($k = 4$ in the cubic case) and n is the number of piecewise segments in the curve (so that for a closed curve n is also the number of control points or nodes p_i). Cubic B-splines exhibit 2nd-derivative continuity between segments. The $\underline{p}_i$'s in Equation 1 are the vertices of a set of control points, which may be linked by linear segments to form a polygon. Associated with each of these vertices is a polynomial function, cubic in this case. The functions N are called basis or weighting functions. For a cubic B-spline, each basis function is a cubic polynomial function in the parameter u. Closed curves are modeled by periodic B-splines; that is, the basis functions are periodic. There will be one such function for every control point $\underline{p}_i$. A curve of given order and given set of basis functions is uniquely determined by the location of the vertices of its associated polygon. Control over curve shape is obtained by manipulating the location of these control points.

We may express Equation (1) using alternative notation:

$$\underline{P}(u) = \begin{bmatrix} u^3 & u^2 & u & 1 \end{bmatrix} B \begin{bmatrix} x_0 & y_0 & z_0 \\ x_1 & y_1 & z_1 \\ x_2 & y_2 & z_2 \\ . & . & . \\ . & . & . \\ . & . & . \\ x_{n-1} & y_{n-1} & z_{n-1} \end{bmatrix} \qquad (2)$$

where B is a 4-by-n matrix, the columns of which contain the coefficients of the basis function associated with each triple $\begin{bmatrix} x_i , y_i , z_i \end{bmatrix}$.

The $\begin{bmatrix} x_i , y_i , z_i \end{bmatrix}$ can be considered as the vertices of the control polygon. There are n of them; each corresponding to a single segment of a piecewise cubic spline.

The equation for each segment of a periodic (closed) B-spline curve can be expressed using similar notation [12]. First, the parameter u is normalized to the range (0,1) as follows:

$$s = \frac{u - u_i}{u_{i+1} - u_i}$$

The i^{th} segment can then be denoted by

$$P_i(s) = [s^3 \quad s^2 \quad s \quad 1]\ \frac{1}{6} \begin{bmatrix} -1 & 3 & -3 & 1 \\ 3 & -6 & 3 & 0 \\ -3 & 0 & 3 & 0 \\ 1 & 4 & 1 & 0 \end{bmatrix} \begin{bmatrix} V_{i-1} \\ V_i \\ V_{i+1} \\ V_{i+2} \end{bmatrix} \tag{3}$$

By substituting 0 and 1 respectively for s in Equation 3, we may obtain expressions for the beginning and ending points of the i^{th} segment in terms of the neighboring vertices of the control polygon. Similar calculations are used to obtain the first and second derivatives of a curve at a given point in parameter space.

B-spline curves exhibit a number of attractive properties from the standpoint of design [13, 14].

(1) They are parametrically defined, enabling the representation of closed curves and other multiple-valued shapes.

(2) They are independent of the coordinate axis system employed.

(3) They are variation-diminishing; that is, first they approximate linear functions exactly, and second, the number of intersections of an approximating curve with an arbitrary straight line is no more than the number of intersections of the primitive function with that line. Intuitively, the approximation will tend to be "smoother" (have fewer undulations) than the actual function.

(4) Because B-spline curves have basis functions which are non-zero through only a portion of their range, they enable local control of curve shape.

(5) Each point on a B-spline curve of order k is a convex combination of control points, and there are at most k points which determine a point on the curve. Thus, a given point on a B-spline curve of degree k-1 will lie in the convex hull of the neighboring k control points, and all points on the curve must lie within the union of all such convex hulls.

(6) When k successive vertices are collinear, they will fully determine one span of the curve. Together with the variation-diminishing property of B-splines, this assures that locally linear segments may be smoothly embedded in a curve.

(7) Corners, cusps, and other discontinuities may be introduced easily without affecting the shape of distant parts of the curve, by using multiple control points; i.e., placing several control points at exactly the same location. The number of control points at a location is called the multiplicity of that point. For a cubic B-spline, a multiplicity of 3 at a control point will produce a curve which passes through that point, and a multiplicity of 3 at two successive control points will yield a smoothly embedded linear segment which is coincident with the corresponding span of the control polygon. It is by means of such embedded linear segments, in fact, that this system represents objects such as cubes.

(8) Increasing the degree of a polynomial approximation in general makes that curve more difficult both to calculate and to control. Through the use of control points and basis functions, B-splines increase the degrees of freedom of the system without necessarily increasing the degree of the curve.

The system

The present version of the system is coded in C and Pascal and runs on a

VAX-11/780 under UNIX.* System modules have monitors inserted in the code for run time profiling. The relational data management system INGRES [15] serves as the back end. Communication with INGRES is through a pre-compiled query-language interface embedded in a C module.

It is envisioned that such a system will eventually fit into the context of a computer-aided design system consisting of applications programs for simulation, analysis, graphic display, and the like, all making use of a general-purpose relational database manager and a common logical data schema.

Data management

The database schema consists of one relation indicating the curves associated with each object, and one emulating a simple linked list which contains the coordinates of control polygon vertices. The schema is as follows:

OBJECTS (OBJID , CURVE1ID, CURVE2ID

CURVES (CURVEID, POINT , X, Y, Z, NEXTPT)

Various non-geometric information about an object, such as material, color, etc., could be maintained in other relations in the schema. It should be noted that such a radically simplified representation was made initially for purposes of analyzing system performance when secondary storage accesses were kept to an absolute minimum. One topic for further research is the effect of incrementally increasing the complexity of the database while reducing the complexity of the geometric-manipulation routines.

We may reconstruct any B-spline curve, which may consist of many piecewise segments, by retrieving the coordinates of its associated polygon. In our first implementation, all objects in the system are defined as pairs of closed-spline curves with linear interpolation between them. The retrieval of an object therefore consists simply of retrieving the control polygon coordinates used to generate its two end curves. This approach appears to be both computationally and storage efficient.

However, if we relax our original requirement that the interpolation between pairs of splines be linear, and instead represent objects by means of a biparametric closed-spline surface, then each surface would be represented in the database as a quasi-cylindrical mesh of control points, with far more complex connectivities between points of the mesh. In this formulation the data structure/schema definition becomes more complex, and manipulative operators more involved, with the countervailing advantage that a larger class of objects, most notably true spheres, may be represented. We are concerned whether the added generality resulting from this approach will justify the increased computation and storage requirements.

* UNIX is a Trademark of Bell Laboratories.

REFERENCES

(1) Blackburn, C. L., Storaasli, O. O., and Fulton, R. E., "The Role and Application of Data Base Management in Integrated Computer Aided Design," presented at AIAA/ASCE/AHS 23rd Structures, Structured Dynamics, and Materials Conference, New Orleans, LA, May 10-12, 1982.

(2) Fishwick, P. A. and Blackburn, C. L., "The Integration of Engineering Programs Using a Relational Database System," presented at 2nd International Computer Engineering Conference and Show, San Diego, CA, August 1982.

(3) Baer, Adrian, Eastman, Charles M., and Henrion, Max, "Geometric Modeling: A Survey," **Computer Aided Design 11**, 5 (September 1979), 253-272.

(4) Requicha, Aristides A. G., "Representation for Rigid Solids: Theory, Methods, and Systems," **Computing Surveys 12**, 4 (December 1980), 437-464.

(5) Boyse, John W., and Gilchrist, Jack E., "GMSolid: Interactive Modeling of Design and Analysis of Solids," **IEEE Computer Graphics and Applications 2**, 2 (March 1982), 27-40.

(6) Hillyard, Robin, "The Build Group of Solid Modelers," **IEEE Computer Graphics and Applications 2**, 2 (March 1982), 43-52.

(7) Brown, Christopher M., "PADL-2: A Technical Summary," **IEEE Computer Graphics and Applications 2**, 2 (March 1982), 43-52.

(8) **Digital Representation for Communication of Product Definition Data**, ANSI Y26.14M, American Society of Mechanical Engineers, New York, 1982.

(9) Faux, I. D. and Pratt, M. J., **Computational Geometry for Design and Manufacture**, Halstead Press, New York, 1980.

(10) de Boor, Carl, **A Practical Guide to Splines**, Springer-Verlag, New York, 1978.

(11) Gordon, William J. and Riesenfeld, Richard F., "B-Spline Curves and Surfaces," in Barnhill, Robert E. and Riesenfeld, Richard F., eds., **Computer Aided Geometric Design**, Academic Press, New York, 1974, 95-126.

(12) Wu, Sheng-chuan, Abel, John F., and Greenberg, Donald P., "An Interactive Computer Graphic Approach to Surface Representation," **CACM 20**, 10 (October 1977), 703-712.

(13) Newman, William M. and Sproull, Robert F., **Principles of Interactive Computer Graphics, 2nd Edition**, McGraw-Hill, New York, 1979.

(14) Rogers, David F. and Adams, J. Alan, **Mathematical Elements for Computer Graphics**, McGraw-Hill, New York, 1976.

(15) Stonebraker, Michael, Wong, Eugene, Kreps, Peter, and Held, Gerald, "The Design and Implementation of INGRES," **ACM Transactions on Data Systems 1**, 3 (September 1976), 189-222.

INTERACTIVE GEOMETRY MODELING OF SPACE STATION CONCEPTUAL DESIGNS

Dariene D. DeRyder, Melvin J. Ferebee, Jr., and Mark L. McMillin
NASA Langley Research Center
Hampton, Virginia

With the advent of a serious interest in the design and implementation of a low-Earth orbit, manned space station, the need has arisen to rapidly synthesize, characterize, and analyze many conceptual designs. Because the manual generation and analysis of a mathematical design model could consume many man-months, an interactive modeling and analysis capability is sought. This poster session will demonstrate the activities and current capabilities at NASA LaRC that could support the modeling analysis of space station conceptual designs.

Previously, geometry modeling of space structures has been done primarily as "stick" or wire-frame type models created with either mathematical synthesizers or finite-element model generators. This method has been adequate for truss-type structures such as antennas or platforms. However, space station concepts now being considered contain solid modules, such as cylinders, consisting of surfaces covered with plates. These concepts, by definition, cannot be modeled as stick elements. The most direct method of modeling these concepts would be to use a solid modeling package. It is believed that solid modeling techniques as well as their implementation into production software packages are in their technological infancy; therefore, it was decided that the purchase of a package at the present time would be premature. Computer Sciences Corporation (CSC) personnel undertook the task of investigating and developing solid modeling software for the purpose of learning solid modeling techniques.

The package developed by CSC has been used for space station conceptual design and visualization. The solid modeling package has not yet been developed to a point that models can be easily converted into finite-element models for analysis programs; therefore, after the initial space station concepts are created as solid models, they must then be created as finite-element models that can be analyzed. The model generated can be visualized as a solid either in the solid modeler or in the MOVIE.BYU software package developed by Brigham Young University.

Because of the inability to readily convert the solid model into a finite-element analysis model, other available geometry modelers were investigated. The PDA Engineering PATRAN-G geometry modeler, which was already available at LaRC, was selected due to its ability to convert the geometry model into a finite-element model. Initially, PATRAN was used to create the space station geometry as one unit, even though the real space station would be modularized. This method of geometry creation proved to be cumbersome when modeling detailed portions of the model, such as intersections and joints. As the users became more familiar with the software and its capabilities, it was determined that the station could be created as modules, stored, and then assembled as desired. Examples of PATRAN models will be presented.

In addition to using PATRAN, designers are also investigating the feasibility of using the ANVIL 4000 software package to create space station design models.

Three space station concepts have been generated using the CSC solid modeler and PATRAN. They have been analyzed with respect to dynamic structural soundness, thermal soundness, and controllability and their results are visualized and presented. Finally, future needs for solid modeling techniques for the generation, characterization, and analysis of space station concepts are presented.

TRANSLATORS BETWEEN CADD AND SECTION 5 OF THE ANSI Y14.26M STANDARD

R. F. Emnett, W. B. Gruttke, E. G. Houghton, and J. E. Oakes
McDonnell Douglas Automation Company, St. Louis, MO

INTRODUCTION

The American National Standard, Engineering Drawing and Related Documentation Practices, Digital Representation for Communication of Product Definition Data (ANSI Y14.26M-1981) [1] comprises an introduction, three sections corresponding to IGES (Initial Graphics Exchange Specification) Version 1.0, and Section 5, a constructive, relational, language-based representation for geometric and topological entities. This presentation discusses the design and development of two-way translators between Section 5 (herein, ANSI5) and CADD (Computer-Aided Design Drafting).

ANSI5

ANSI5 was developed over several years by participants from academia, government, and industry in the subcommittee Y14.26 and in task group Y14.26.1. The general motivation for the Standard was to facilitate communication between diverse CAD/CAM (Computer-Aided Design/Computer-Aided Manufacturing) systems. The methodology described in ANSI5 is designed to provide representations for basic shape data utilizing a minimum number of structures while providing a broad geometric scope. The constructive representation allows this design goal to be realized by use of the same structures to represent curves, surfaces, and solids. The language representations of ANSI5 provide a concise description of the geometric and topological entities by relationally grouping entities of the same structure. Thus, the overhead necessary for structure description need not be repeated with each similarly defined entity. The historical development and geometric coverage of ANSI5 are presented in Figure 1.

CADD

CADD was developed by McDonnell Douglas Corporation (MDC) in the late 1960's. It is a fully three-dimensional design package implemented in both distributed graphics and mainframe configurations. It can manipulate models up to 900K bytes in size. Capabilities include creation, deletion, and replication of 28 entity types of wire frame geometry, surface geometry, annotation, and groups.

DESIGN

The CADD-to-ANSI5 translator is divided into three functional modules:

- o Batch Retrieval - CADD utility to retrieve the model from disk
- o Entity Conversion - entity by entity conversion to the ANSI5 mathematical format and generation of ANSI5 language statements
- o Imbedding in File Structure - imbeds the ANSI5 language statements in the file structure described in Section 2 of the Standard

The output is an ANSI5 model for translation to another modelling system format.

The ANSI5-to-CADD translator consists of four modules:

- o ANSI5 Language Extraction - extracts ANSI5 statements from the file structure
- o ANSI5 Language Parser - parses all statements, identifies syntax errors, and stores the data in arrays
- o Entity Conversion - uses the output of the parser to create appropriate CADD entities
- o Batch Filing - CADD utility which stores the model

In the parser and ANSI5-to-CADD entity conversion modules, one ANSI5 statement at a time is parsed and then used to create CADD entities.

An alternate architecture was identified for cases where imbedding the statements in the file structure is not desired. The two entity conversion modules and the parser form the bulk of the design and are discussed further. The two architectures are represented by Figures 2 and 3.

As for all pairs of distinct geometry representation systems, the entities in CADD and in ANSI5 are not identical. Therefore, not all entities will translate directly from one system to the other. In some cases, the point set represented by the entity can be preserved, but not the parameterization. In other cases, only approximation is possible. The translation options available for any two geometric representation systems are summarized in Figure 4. Figure 5 reflects the options pertinent to the CADD/ANSI5 translators. Entities are translated directly whenever possible, preferably retaining the parameterization. Otherwise, an approximation is attempted. Curves are approximated by a cubic spline; surfaces by a group of boundary curves. If none of these options is applicable, the entity is not translated. ANSI5 addresses only geometric entities; therefore the non-geometric CADD entities (e.g., text and dimensioning entities) are not addressed.

The CADD-to-ANSI5 entity conversion module is divided into sub-modules by CADD entity type. The CADD entities and the corresponding ANSI5 structures are presented in Figure 6 in the order processed. All CADD entities are translatable directly into ANSI5 structures, except for unbounded planes and certain non-planar parametric ruled surfaces.

The ANSI5-to-CADD entity conversion module is divided into sub-modules by ANSI5 structure. The correspondence between these structures and the resultant CADD entities is presented in Figure 7. ANSI5 entities which do not translate directly into CADD entities are approximated if possible.

The ANSI5 parser checks language statements according to the syntax rules defined in ANSI5. The semantics of a given structure are transparent to the parser; therefore, adding new ANSI5 structures or redefining existing structures would not necessitate changing the parser.

PROBLEMS

Aside from the usual problems related to learning curves and incomplete or outdated documentation, three problems were primarily encountered in the design and development of these translators:

- o additional entities are sometimes required to define geometry in ANSI5 format
- o associated entity parameterizations are not always retainable
- o lack of entity correlation between formats forces approximations

Translation of certain entities requires creation of entities not in the original format. This is true of the translators in both directions. An example is that two additional points may be required to describe cubic curve entities in the CADD-to-ANSI5 mode. Conversely, additional boundary curves may be required to translate certain interpolative surfaces in the ANSI5-to-CADD mode.

ANSI5 uses rational parametric geometry in its structures. The implied parameterizations used by CADD do not always coincide with this approach. This is not a particular problem for the curves themselves. However, this difference may affect surfaces generated using these curves.

Approximation arises from the lack of exact correspondence between the CADD and ANSI5 entities. Some examples are the ANSI5 N-th degree polynomial structure (approximated by cubics in CADD), certain ANSI5 interpolative surfaces (some approximated and others not representable in CADD), and the CADD unbounded plane (approximated by a "very large" rectangle in ANSI5).

PERFORMANCE

Six CADD file parts (as depicted in Figures 8 throuth 13) were studied for this paper. Performance data for comparison between CADD/ANSI5 translators and CADD/IGES processors are presented. The number of each type of entity and the model size for each of the six parts are presented in Figure 14.

The six parts are presented only as wire frame geometry. The CADD-to-ANSI5-to-CADD translators and the CADD-to-IGES-to-CADD processors were used for all six parts. The model sizes and processing times are presented in Figures 15 and 16. The final entity counts and model sizes produced in each case are contained in Figure 17. Figure 18 presents some observations about the preceding statistics.

SUMMARY

In summary, ANSI5 provides a viable representation system for communicating three-dimensional models. The design and development of translators between CADD and ANSI5 presented no major obstacles. The performance of the CADD/ANSI5 translators compares favorably with that of the CADD/IGES processors.

REFERENCES

1. Digital Representation for Communication of Product Definition Data. ANSI Y14.26M-1981, American Society of Mechanical Engineers, 1982.

2. Design of an Experimental Boundary Representation and Management System for Solid Objects. Report No. R-80-GM-02, Computer Manufacturing International, Inc., 1980.

STRUCTURE TYPE	ANSI Y14.26.1 - 1978*	CAM-I GMP - 1980 [2]	ANSI Y14.26M(5) - 1981 [1]
G E O M E T R I C			
PRIMITIVE			
Point	✓	✓	✓
Circle	None	Parameterization Not Specified	Parameterization Specified
INTERPOLATIVE			
Linear	✓	✓	✓
Circular Arc	✓	Fixed Parameterization	Parameterized by User
Conic Arc	✓		
Cubic Arc	✓		
Spline	None	B-Spline	✓
N-th Degree Polynomial	None	None	✓
GENERATIVE			
Translation/Rotation	✓	✓	Improved Definitions
Rotation	✓	✓	Improved Definitions
Translation	✓	✓	✓
SPECIAL			
Reverse	None	✓	✓
Curve String	No Reparameterization	Reparameterized	Definition Modified
Evaluation	None	✓	✓
Flip	None	✓	✓
POINT SET			
Dimension Selecting Closure	None	None	✓
Union	Presented But Definitions Not Well-Defined	Rigorous Definitions, Selectivity	Connected Selectivity Dropped
Intersection			
Difference			
Cross Intersection		None	None
Non-Cross Intersection		None	None
REPLICATIVE			
Translation/Rotation	✓	Major Notation Revision to Improve Clarity	✓
Rotation	✓		✓
Translation	✓		✓
Scale	✓		✓
Reflection	✓		✓
T O P O L O G I C A L			
Vertex	None	✓	✓
Edge	None	✓	✓
Switch	None	✓	✓
Loop	None	✓	✓
Face	None	Definition Weak	Definition Rigorous
Shell	None	✓	Definition Modified
Object	None	✓	✓
M I S C E L L A N E O U S			
Group	✓	Changed	Further Modified
U N C L A S S I F I E D			
Object (Non-Topological)	✓	None	None
Object File	✓	None	None
Tuple Deletion	✓	Interactive Query/Edit	None, None } Database Function
Tuple Insertion	✓		
Domain Structure	None	✓	✓

*Superseded by Section 5 of ANSI Y14.26M-1981 [1].

Figure 1.- A brief history.

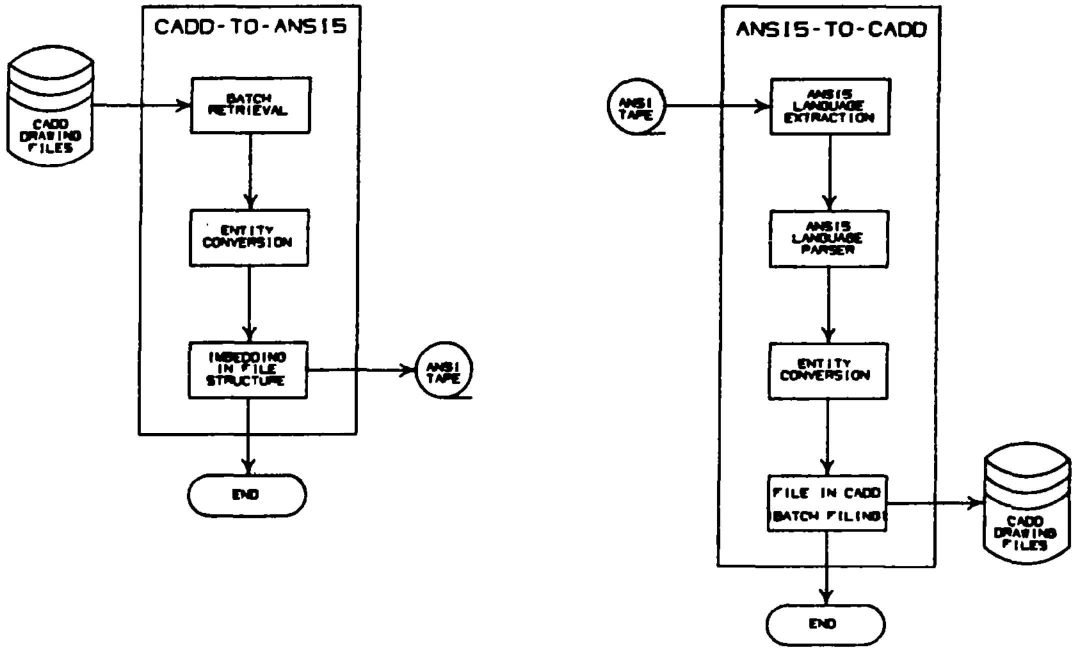


Figure 2.- Architecture.

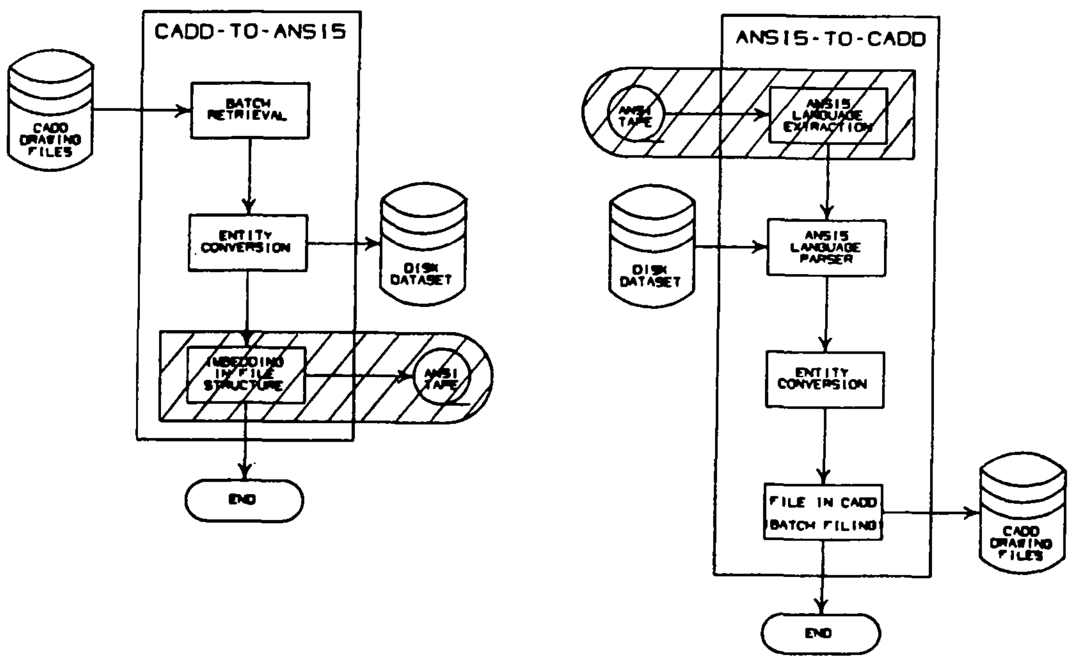


Figure 3.- Alternate architecture.

Original Geometry	Translation Options
Points	- Translate directly - Approximate - Not translatable
Curves	- Translate directly (parameterization retained) - Translate directly (parameterization lost) - Approximate - Not translatable
Surfaces	- Translate directly (parameterization retained) - Translate directly (parameterization lost) - Approximate - Translate as group of curves - Not translatable
Solids	- Translate directly (parameterization retained) - Translate directly (parameterization lost) - Approximate - Translate as group of surfaces - Translate as group of curves - Not translatable

Figure 4.- Translation options.

Original Geometry	Translation Options
Points	- Translate directly - Not translatable
Curves	- Translate directly (parameterization retained) - Translate directly (parameterization lost) - Approximate - Not translatable
Surfaces	- Translate directly (parameterization retained) - Translate directly (parameterization lost) - Translate as group of curves - Not translatable
Solids	- Not translatable

Figure 5.- CADD/ANSI5 translation options.

CADD ENTITY	ANSI5 STRUCTURE
Point	Point
Crosshair	Point
Line	Linear
Plane	Linear
Arc	Circular Arc
Circle	Circle
PC Curve	Cubic
Conic	Conic
PC Patch	Cubic
Sphere	Rotation Generation
Bounded Plane	Face
Parametric Ruled Surface	Linear or Group
Group	Group

Figure 6.- CADD-to-ANSI5 entity correspondence.

ANSI5 STRUCTURE(1)	CURVES(2)	SURFACES(2)
GEOMETRIC STRUCTURES		
INTERPOLATIVE		
Linear	Line	PRS(s)(3), BP(3), Curve Group
Circular Arc	Arc	PRS, BP, Curve Group
Conic	Conic	PRS, BP, Curve Group
Cubic	Cubic	PC-Patch(s)(3), BP, Curve Group
Nth Degree Polynomial	Line, Parabola, Cubic, Cubic Curve Group	PRS(s), PC-Patch(s), BP, Curve Group
Spline	Cubic Curve Group	PC-Patch(s), BP, Curve Group
GENERATIVE		
Translation/Rotation	Line, Arc, Conic, Cubic, Cubic Curve Group	PC-Patch, Curve Group
Rotation	Arc	PRS(s), BP, Sphere, Curve Group
Translation	Line, Arc, Conic, Cubic, Cubic Curve Group	PRS(s) and/or PC-Patch(s), BP, Curve Group
REPLICATIVE		
Scale	Line, Arc, Conic, Cubic Curve Group	Surface of Same Type
Other	Curve of Same Type	Surface of Same Type
SPECIAL		
Flip	Not Applicable	Surface of Same Type
Reverse	Curve of Same Type	Not Applicable
Evaluation	Not Translated	Not Translated
Curve String	Curve Group	Not Applicable
POINT SET		
Intersection	Group	Not Applicable
Dimension-Selecting Closure	Curve Group	Not Applicable

(1) Structures not listed either are not translated or the translations are obvious
(2) Some specific cases are not translated
(3) PRS = Parametric Ruled Surfaces, BP = Bounded Plane, PC = Parametric Cubic

TOPOLOGICAL STRUCTURES	TRANSLATED CADD ENTITY
Vertex	Point
Edge	Curve
Switch	Curve
Loop	Group of Curves
Face	Surface, Bounded Plane, or Can't Be Modelled
Shell	Group of Surfaces and Bounded Planes
Object	Not Modelled or as Above
MISCELLANEOUS STRUCTURE	
Group	CADD Group

Figure 7.- ANSI5-to-CADD entity correspondence.

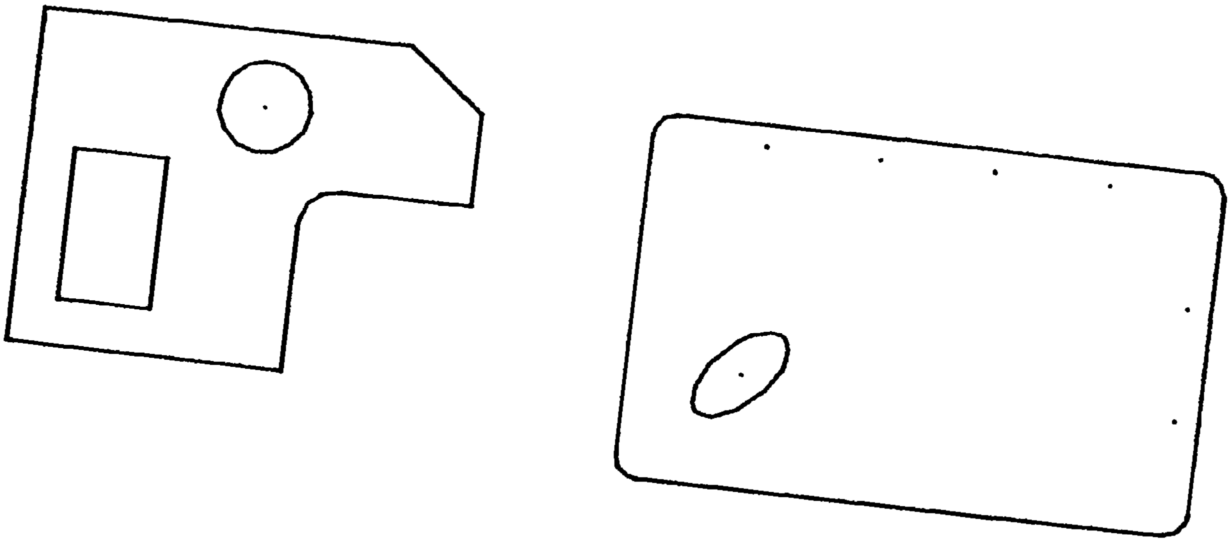


Figure 8.- Simple 2-D part.

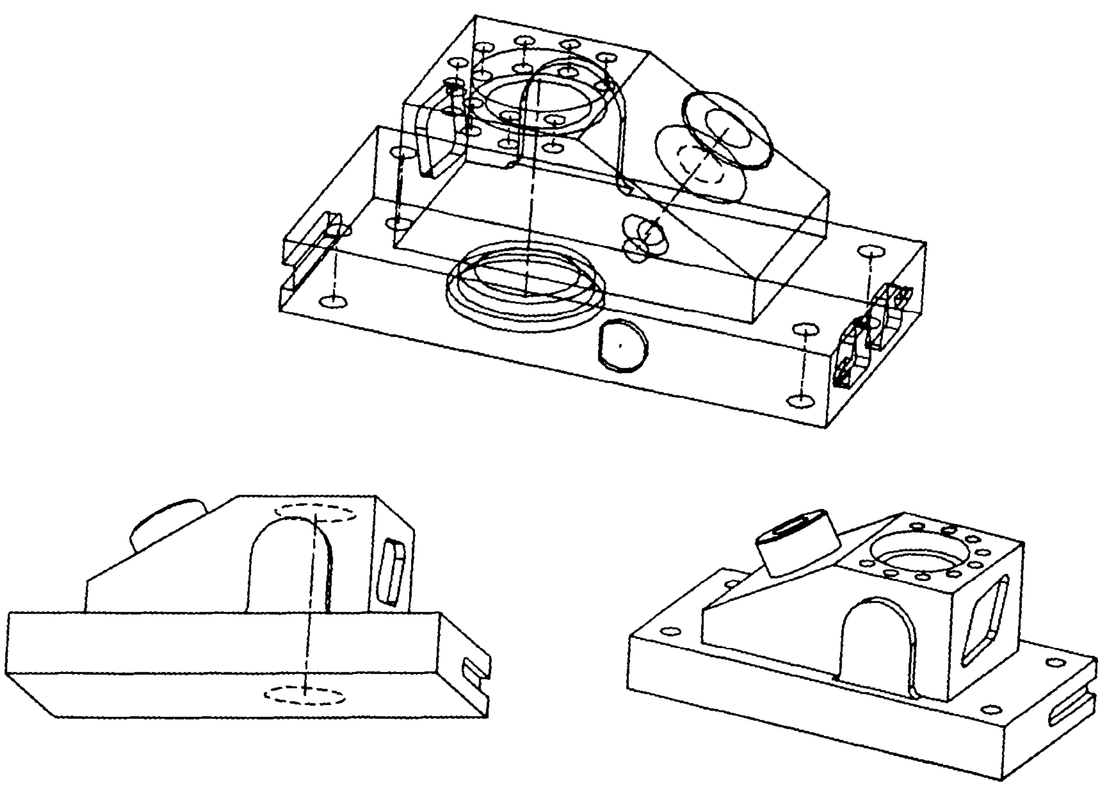


Figure 9.- ANC-101.

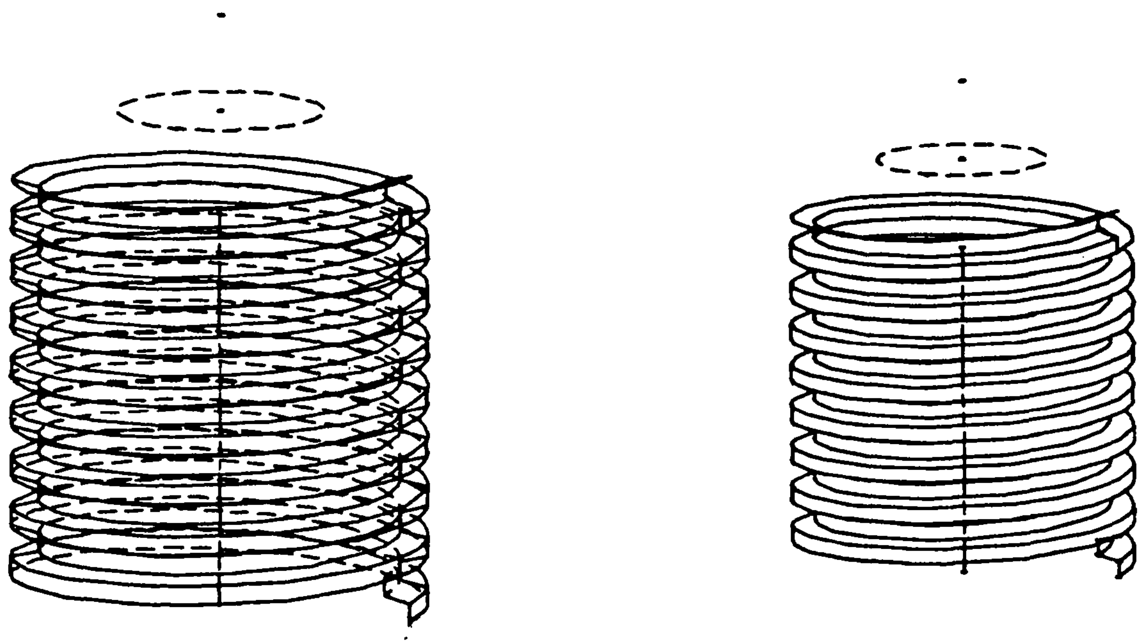


Figure 10.- Bolt thread.

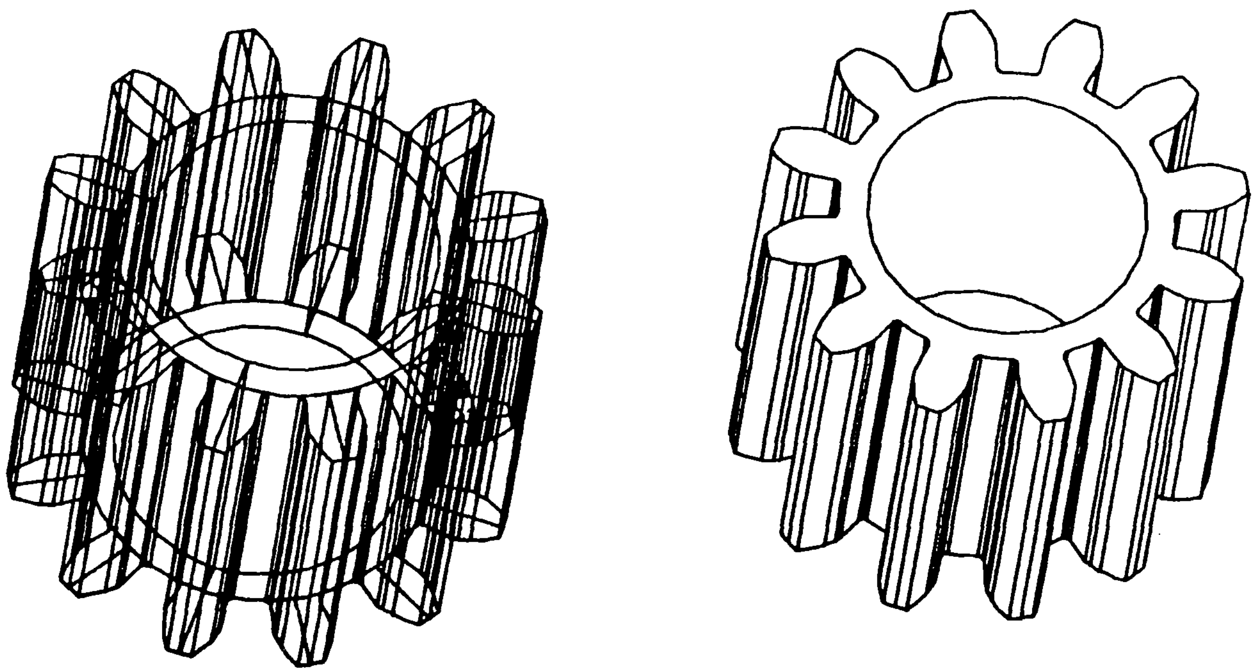


Figure 11.- Gear.

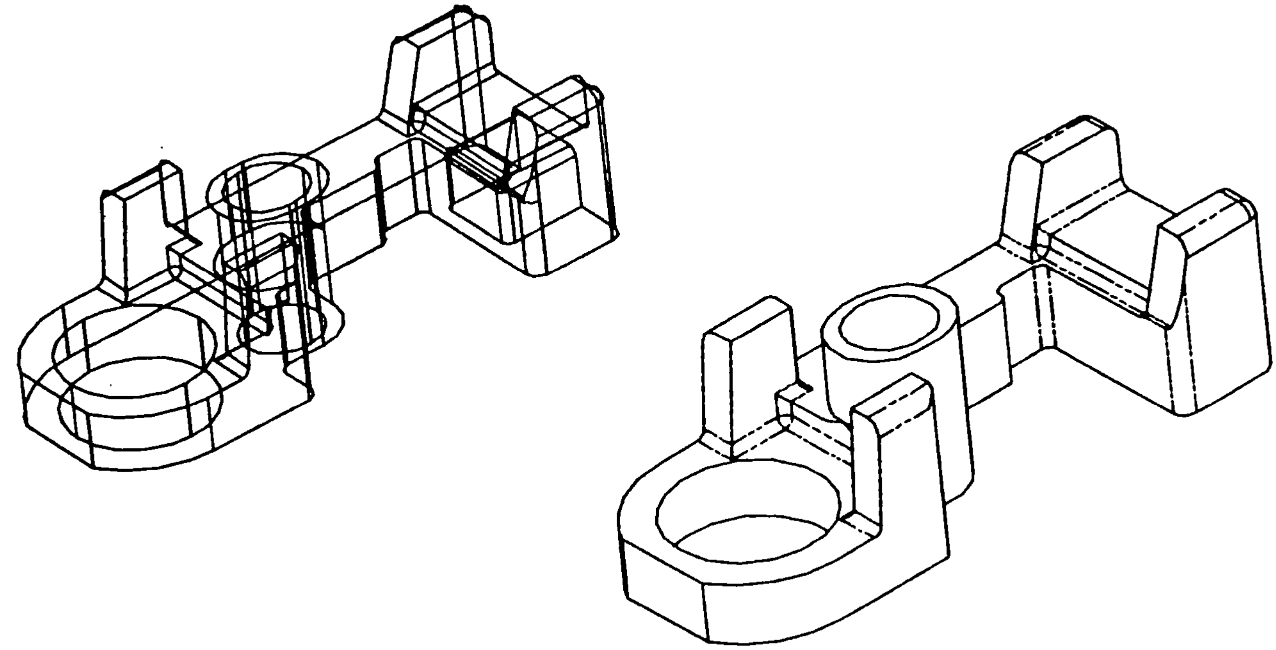


Figure 12.- CAM-I GMP part.

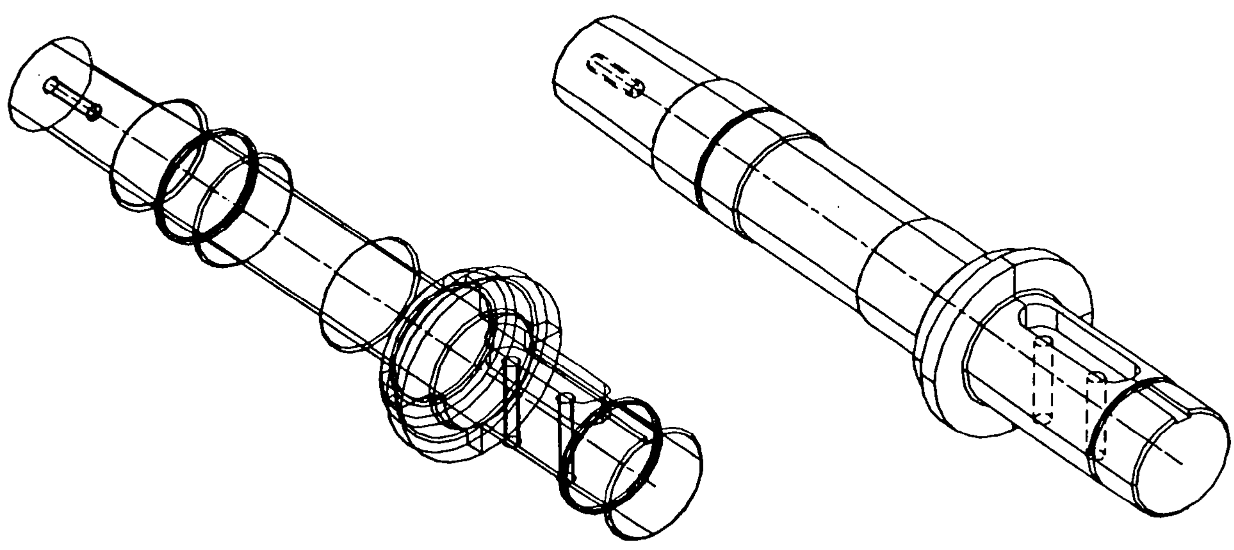


Figure 13.- Turned part.

PART TO BE TRANSLATED	CADD FILE SIZE (BYTES)	NUMBER OF ENTITIES POINT	CURVES LINE	ARC/ CIRCLE	CONIC	PC	TOTAL
SIMPLE 2-D	2,520	30	15	6	1	-	52
ANC101	17,856	140	138	80	-	-	358
BOLT THREAD	34,864	299	73	1	-	201	574
GEAR	39,392	392	330	148	-	-	870
CAM-I GMP	21,724	168	160	58	-	55	441
TURNED	34,788	171	17	2	-	287	477

Figure 14.- Part before translation.

PART/TRANSLATOR		CHARACTER COUNT		PROCESSING TIME (MIN)			
		LANGUAGE	FILE	TO DISK CPU	TO DISK I/O	TO TAPE CPU	TO TAPE I/O
SIMPLE 2-D	ANSI5	3,093	5,280	.017	.204	.023	.255
	IGES	N/A	10,640	N/A		.026	.324
ANC101	ANSI5	18,944	24,960	.027	.212	.035	.277
	IGES	N/A	71,840	N/A		.064	1.091
BOLT THREAD	ANSI5	45,615	58,400	.066	.226	.081	.312
	IGES	N/A	159,280	N/A		.129	2.188
GEAR	ANSI5	36,708	47,120	.042	.224	.053	.295
	IGES	N/A	153,840	N/A		.131	2.120
CAM-I GMP	ANSI5	23,952	31,200	.030	.215	.039	.284
	IGES	N/A	107,040	N/A		.086	1.532
TURNED	ANSI5	62,691	80,000	.073	.228	.089	.319
	IGES	N/A	191,680	N/A		.146	2.593

Figure 15.- CADD-to-XXXX translators statistics.

PART/TRANSLATOR		CHARACTER COUNT		PROCESSING TIME (MIN)			
		LANGUAGE	FILE	FROM LANGUAGE		FROM TAPE	
				CPU	I/O	CPU	I/O
SIMPLE 2-D	ANSI5	3,093	5,280	.025	.221	.037	.305
	IGES	N/A	10,640	N/A		.073	.862
ANC101	ANSI5	18,944	24,960	.111	.224	.123	.308
	IGES	N/A	71,840	N/A		.316	3.994
BOLT THREAD	ANSI5	45,615	58,400	.323	.242	.335	.328
	IGES	N/A	159,280	N/A		.554	6.925
GEAR	ANSI5	36,708	47,120	.259	.241	.269	.310
	IGES	N/A	153,840	N/A		.650	8.307
CAM-I GMP	ANSI5	23,952	31,200	.133	.232	.145	.316
	IGES	N/A	107,040	N/A		.408	5.449
TURNED	ANSI5	62,691	80,000	.303	.264	.316	.348
	IGES	N/A	191,680	N/A		.589	7.805

Figure 16.- XXXX-to-CADD translators statistics.

PART		CADD FILE SIZE	NUMBER OF ENTITIES					
				CURVES				
			POINT	LINE	ARC/ CIRCLE	CONIC	PC	TOTAL
SIMPLE 2-D -	BEFORE	2,520	30	15	6	1	-	52
	ANSI5	3,456	46	15	6	2	-	69
	IGES	2,676	33	15	6	1	-	55
ANC101 -	BEFORE	17,856	140	138	80	-	-	358
	ANSI5	24,460	267	138	80	-	-	485
	IGES	17,856	140	138	80	-	-	358
BOLT THREAD -	BEFORE	34,864	299	73	1	-	201	574
	ANSI5	55,716	700	73	1	-	201	975
	IGES	34,864	299	73	1	-	201	574
GEAR -	BEFORE	39,392	392	330	148	-	-	870
	ANSI5	44,592	492	330	148	-	-	970
	IGES	39,392	392	330	148	-	-	870
CAM-I GMP -	BEFORE	21,724	168	160	58	-	55	441
	ANSI5	29,628	320	160	58	-	55	593
	IGES	21,776	169	160	58	-	55	442
TURNED -	BEFORE	33,696	171	17	2	-	287	477
	ANSI5	63,180	738	17	2	-	287	1,044
	IGES	33,696	171	17	2	-	287	477

Figure 17.- Part after translation (CADD-XXXX-CADD).

- Computing Environment: IBM 3081 - MVS
  95% FORTRAN - 5% Assembler

- 6 sample parts varying significantly in content and complexity

- For CADD-ANSI5-CADD: Two sets of processing times reflecting the dual architectures

- ANSI5 File to IGES File average character count ratio:
  ANSI5 imbedded in IGES: 37.1%
  ANSI5 language only: 27.2%

- ANSI5 to IGES average processing time ratios:
  CPU minutes: 56.2%
  I/O minutes: 14.4%

- Large numbers of extra points generated by ANSI5 cycle are all simply removable from CADD.

- ANSI5 and IGES processors remain in development and are still being tuned.

Figure 18.- Comments on statistics.

AN INTERSECTION ALGORITHM FOR MOVING PARTS*

D. M. Esterling
Joint Institute for Advancement of Flight Sciences
George Washington University
NASA Langley Research Center
Hampton, Virginia

J. Van Rosendale
Institute for Computer Applications in Science and Engineering
NASA Langley Research Center
Hampton, Virginia

The problem of deciding whether moving mechanical parts will collide during motion arises frequently in computer-aided design of machinery and in robotics. In machinery design, the problem becomes acute when there are various complex curved mechanical parts that will follow planned trajectories. There is a need to know at an early stage in the design process whether there will be collisions between parts necessitating changes in part shapes, trajectories, or in the overall design.

Collision detection is also an important problem in robotics when there is a need to know whether a planned trajectory for a robot arm is acceptable. Unplanned collisions could damage parts being handled or the robot arm itself and could create safety hazards. Motion planning can be done either by a human operator or by a heuristic computer algorithm. In either case, a reliable collision detection algorithm is needed. When a collision will occur, it is desirable to know the point of first occurrence. The human operator or motion planning algorithm can use this information to attempt construction of a better trajectory or to modify the parts.

Whether collision detection is done for robot motion planning or for computer-aided machinery design, reliability of the collision detection algorithm is paramount. An algorithm that detects some collisions but misses others is virtually useless. The efficiency of the algorithm is also important, especially in robot motion planning where mini- or micro-computers are often employed.

This paper describes a collision detection algorithm that is guaranteed to detect a collision if one will occur, and it also computes the earliest point of contact. Earlier work by Carlson (ref. 1) developed an efficient algorithm for finding the intersection of a pair of stationary bicubic patches. Schwartz (ref. 2) has described a method for finding the minimum distance between two moving polygons. Our algorithm applies to any solid geometry parts model of the boundary representation type; that is, solids are described by the list of curved parametric surfaces which bound them. Completely general, curved trajectories are allowed as well.

The basis of this algorithm is a restatement of the moving parts collision problem as a surface intersection problem in four-dimensional space time. This

*This research was supported by the National Aeronautics and Space Administration under NASA contracts NCC1-36, NAS1-15810 , and NAS1-16394.

surface intersection problem is approached via a simple subdivision algorithm yielding the desired reliability. The computational cost of this algorithm is relatively modest since the subdivision is done by a depth first tree search that computes only the earliest point of contact between the moving parts.

In the boundary representation scheme, each part is represented by a list of boundary surfaces, and each surface is defined by a parametric mapping (fig. 1). In this figure, the unit square in parameter space maps into a general surface S in R^3. One way to compute the intersection of two such parametric surfaces is to use subdivision. In a subdivision algorithm, one recursively subdivides the parameter space for a surface into small cells that may contain points of intersection. Cells away from the intersection are discarded in this process; those near it are defined until a tolerance criterion is met. Figure 1 shows such a cell in parameter space and its image in R^3. The image patch is covered by a spherical neighborhood that is used to detect intersections. If the spherical neighborhood of a patch overlaps the neighborhood of some patch on another surface, these two patches may intersect. Those patches in which spherical neighborhoods meet no neighborhoods on the other surface contain no intersection points and can be eliminated from further consideration.

The algorithm here is based on applying this subdivision idea in four-dimensional space time. To see how this would work, consider a lower dimensional analog, a line segment moving in the x-y plane. Figure 2 shows a parametric representation of such a moving line segment. On the right is the space-time image of the moving line segment. On the left is the corresponding parameter space. Here the parameter τ maps one-to-one into the time interval of interest. We may define cells in parameter space here as shown in figure 2 just as for the motionless surface in figure 1. The image of a rectangular cell in parameter (r, τ) space is a curved patch in three-dimensional (x, y, t) space time. This curved patch may be covered by a space-time cylinder as shown.

These cylindrical space-time neighborhoods can be used to compute the intersection of the space-time images of two moving objects; that is, they can be used to detect collisions between moving objects. To do this, we carry out refinement in the parameter space of each object always looking first for collisions at the earliest time since only the first collision point is required. Each outer loop in the refinement process starts with a refinement on time (or the parameter τ) for every surface. Next, a spatial tolerance level (tolsp) is set in R^3 which is proportional to the current time interval. Within this time interval, each surface is divided into subregions or neighborhoods in R^3, and the neighborhoods are tested for overlap with cells on the other surfaces.

In the refinement process, a list of the cells which have a corresponding overlap is kept for each surface. These cells will then be the candidates for further subdivision. The situation at a given time refinement level for another surface is depicted in figure 3. The object here is a circle moving in the x-y plane. Cells in the (r, τ) parameter space map to corresponding images in (x, y, t) space time as shown. Cells in which cylindrical space-time neighborhoods overlap the space-time neighborhoods of cells on the other surface will be kept for further refinement. The others will be deleted.

Figure 4 shows two circles, which never collide, moving in the plane. The subdivision process finds there is no collision at a relatively modest computational cost (depending on how close they approach each other). The space-time refinement process concentrates the computational effort in the region of space time where the objects are closest, as shown. This is the advantage of the approach described here.

A related concept that does not carry the adaptive refinement idea quite as far is described in Schwartz (ref. 2) for polygonal moving parts in two dimensions.

Though the examples given have been of two-dimensional objects moving in the plane, the case of three-dimensional objects moving in space works in a similar manner. The neighborhoods covering the space-time cells become four-dimensional cylinders formed by moving a stationary sphere through a time interval. Deciding whether two of these four-dimensional cylinders overlap is mathematically trivial, so this algorithm is relatively efficient with the bulk of the computational effort devoted to data structure computations.

The data structures for this algorithm have been programmed in PASCAL. Space does not permit a careful description. However, we wish to point out one of its main features. At each stage in the refinement process, there will be a number of neighborhoods covering the possible collision points on each moving part. To perform overlap tests for each neighborhood on one part with every neighborhood on the other part is highly inefficient. This is avoided here by maintaining lists of pointers telling which parts of neighborhoods may overlap.

The algorithm described is completely general and relatively efficient. No constraints on the type of parts or their trajectories are imposed by the method. The method is perfectly reliable and always detects a collision if one occurs. The simple and fast numerical tests involved, together with a carefully designed data structure that maintains lists of active overlapping cells, yield an effective algorithm for moving parts collision detection.

REFERENCES

1. Carlson, W. E.: An Algorithm and Data Structure for 3D Object Synthesis Using Surface Patch Intersections. Computer Graphics, vol. 16, no. 3, (SIGGRAPH '82), July 1982, pp. 255-259.

2. Schwartz, J. T.: Finding the Minimum Distance Between Two Convex Polygons. ICASE Report No. 81-22, Institute for Computer Application in Science and Engineering, NASA-Langley Research Center, Hampton, Virginia, July 1981.

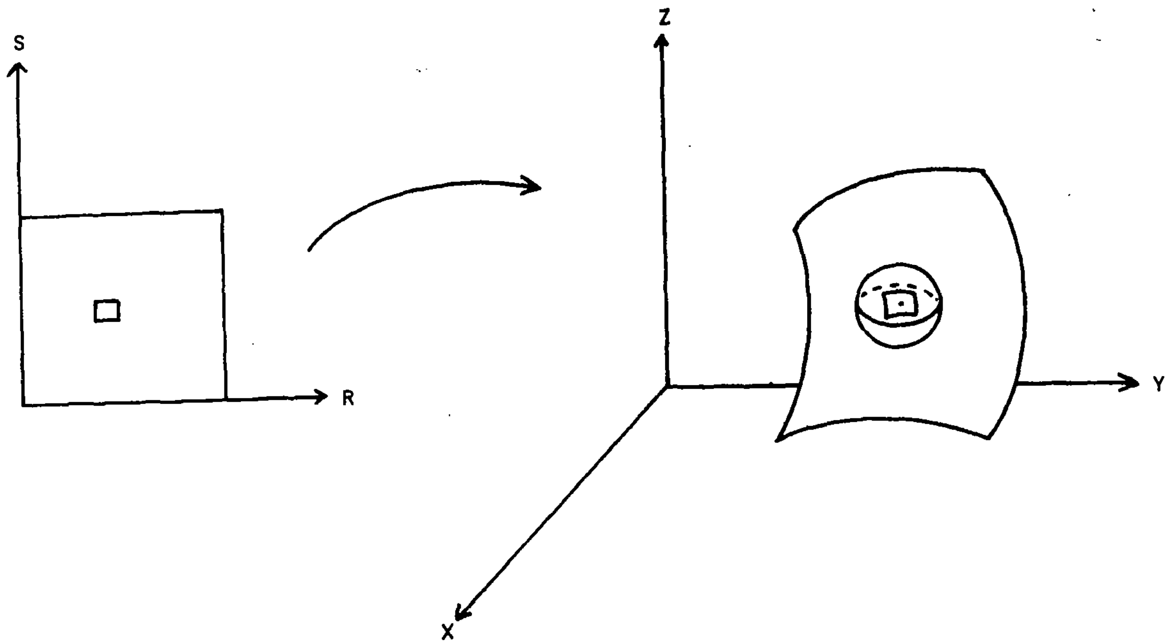


Figure 1.- Parametric mapping and covering spherical neighborhood.

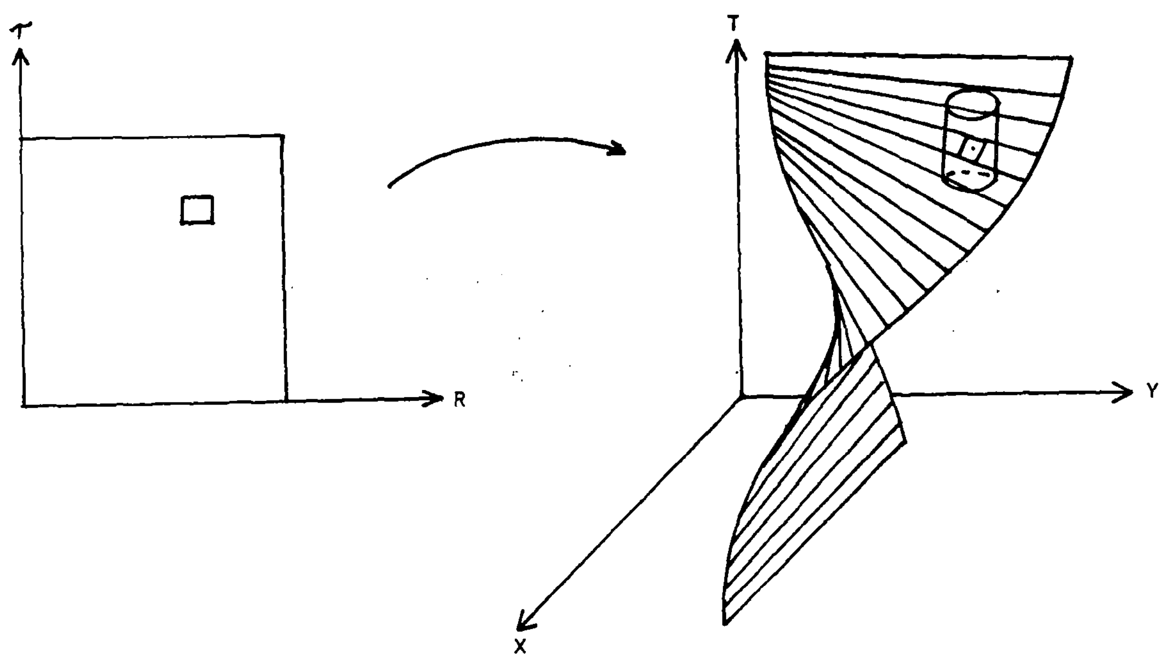


Figure 2.- Covering space-time cylinder for a line segment moving in time.

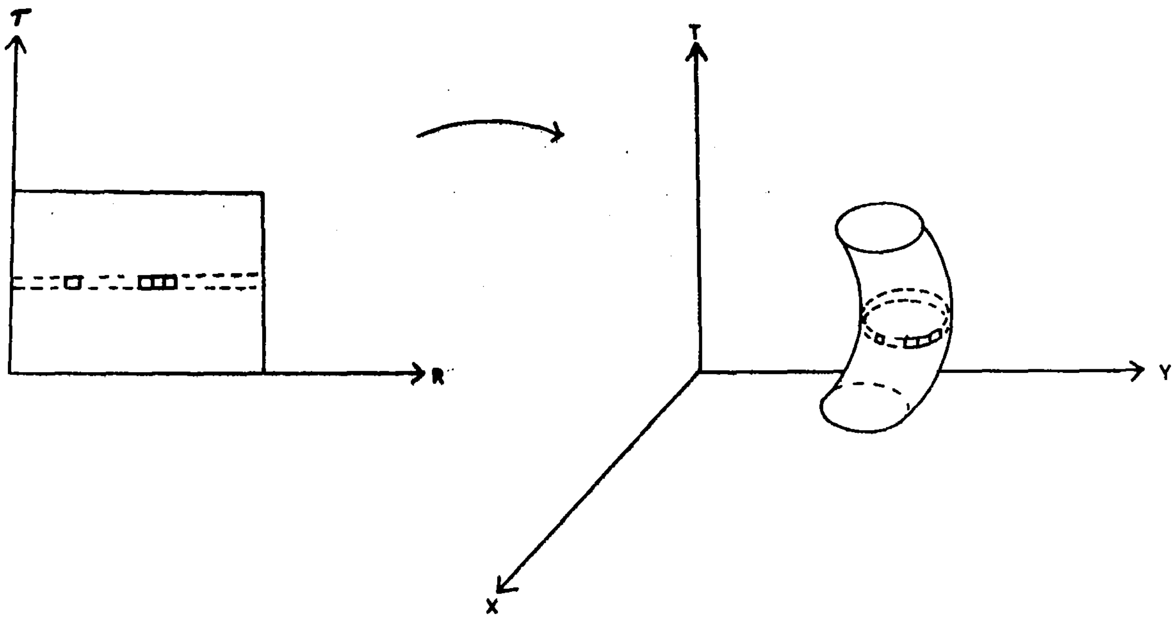


Figure 3.- Cells in a given time interval for a circle moving in time.

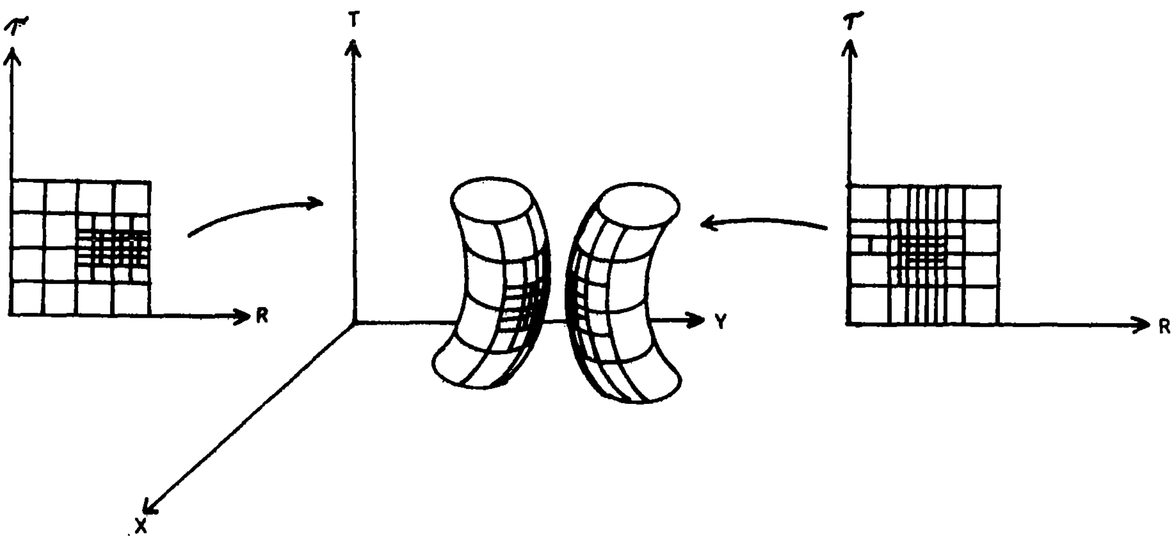


Figure 4.- Adaptive refinement process for nearly colliding circles.

GENERATION OF SURFACE-FITTED COORDINATE GRIDS

A. Garon and R. Camarero

Ecole Polytechnique, Montreal, Canada

INTRODUCTION

An accurate numerical solution of the Navier-Stokes equations is strongly dependent on the description of the flow geometry. In the past, rectangular grids led to an ill description of curved boundaries and consequently hindered finite-difference solutions. Now, it is accepted that body-fitted coordinate systems give a much better description of complex flow geometries.

The generation of body-fitted curvilinear coordinate grids in two dimensions by the solution of a nonlinear system of elliptic equations has been proposed by Thompson (ref. 1). It has been extended to three dimensions by Mastin and Thompson (ref. 2) and applied in practical configurations by Thames (ref. 3), Camarero and Reggio (ref. 4).

In this approach a physical domain (by ext. cartesian) bounded by six surfaces is mapped into a right angle parallelepiped by a one-to-one transformation. The transformation and hence the grid in the physical region are obtained numerically as the solution of a Dirichlet problem where the values of the coordinates are specified on each of the six surfaces. However the resulting three-dimensional grid in the vicinity of the boundaries will depend very much on this choice. This is critical because the accuracy of the solution of the Navier-Stokes equations is directly dependent on the large gradients that prevail in this region. Moreover the use of curvilinear coordinate grids complicate the flow equations and induce geometric errors (ref. 5). This is not an ideal situation and it would be preferable to have the resulting grid depend on the physics of the flow.

An optimal three-dimensional grid is usually found by a painstaking choice of boundary coordinates. These coordinates are themselves surface oriented grids. Then it must be possible to find two-dimensional surface-fitted grid equations. In this approach the surface is to be mapped into a rectangle by a one-to-one transformation. Thomas (ref. 6) has published a method for this problem based on the solution of a nonlinear system of elliptic equations. In fact these equations are a generalization of Thompson's two-dimensional grid equations.

To obtain his grid equations Thomas applied a mathematical restriction of Mastin and Thompson's three-dimensional grid equations. These are found by assuming that the coordinate lines directed out of the surface are normal and have zero curvature. Moreover he postulates the surface to be defined in cartesian coordinates by the function $z = f(x,y)$, where f is single valued and twice-differentiable, $C^{2,2}$. However, this is not always possible globally. Hence the surface must be subdivided in subregions where the choosen parametric equation exists.

In this paper we propose a new set of surface-fitted grid equations based on the one-to-one twice-differentiable parametric equations,

$$x = x(u,v)$$

$$y = y(u,v)$$

$$z = z(u,v)$$

Then a surface-fitted grid is found even if the function $z = f(x,y)$ does not exist globally. The present approach avoids the use of branch cuts and subregions. Moreover if we know the surface by a finite set of points, the present method can be used easily with any $C^{2,2}$ parametric patch generation technique. The present two-dimensional surface-fitted grid equations were not obtained by mathematical restriction. Rather the system of two Poisson equations (originally used by Thompson), written with the metric coefficients induced by the parametric equations above, is simply inverted.

GENERALIZED TWO-DIMENSIONAL GRID EQUATIONS

In the present approach we consider the surface to be represented without restriction by the following choice of parametric equations:

$$P = \begin{cases} x = x(u,v) \\ y = y(u,v) \\ z = z(u,v) \end{cases} \tag{1}$$

with $(u,v) \in U$, a compact subregion of $\mathcal{R}^2$. We expect these single valued functions to be twice-differentiable, $C^{2,2}(U)$. With this parametrization the surface is expected to be defined without ambiguity, meaning that for any point of the surface there exists one and only one point of U and conversely. These are not arbitrary conditions; they ensure continuity of the metric tensor and of the Laplacian operator. Moreover, now we can define a surface-fitted coordinate grid globally.

It is known that at any point of the surface we can define an invariant, the first fundamental quadratic form (ref. 7):

$$ds^2 = g_{11} du^2 + 2\, g_{12} du\, dv + g_{22} dv^2 \tag{2}$$

where ds is the distance between two points infinitely close to the surface. The coefficients, elements of the metric tensor (g_{ij}), are related to the tangent vectors at any point of the surface as follows:

$$\begin{aligned} g_{11} &= \vec{g}_1 \cdot \vec{g}_1 = x_u^2 + y_u^2 + z_u^2 \\ g_{22} &= \vec{g}_2 \cdot \vec{g}_2 = x_v^2 + y_v^2 + z_v^2 \\ g_{12} &= \vec{g}_1 \cdot \vec{g}_2 = x_u x_v + y_u y_v + z_u z_v \\ g_{21} &= g_{12} \end{aligned} \tag{3}$$

We know that the metric tensor is symmetrical and positive definite. Then this matrix possesses an inverse, noted (g^{ij}), with the following elements:

$$\begin{aligned} g^{11} &= g_{22}/g \\ g^{22} &= g_{11}/g \\ g^{12} &= -g_{12}/g \\ g^{21} &= g^{12} \end{aligned} \tag{4}$$

and

$$g = g_{11}g_{22} - g_{12}^2$$

is the determinant of the metric tensor. Now say that we have a single valued function $f(u,v)$, twice-differentiable, representing for argument's sake the temperature at any points of the surface. Then the heat equation is given by a Poisson equation

$$\Delta(f) = \mathrm{div}(\nabla f) \tag{5}$$

To include the geometry of the surface, the Laplacian operator is written by means of the metric coefficients:

$$\Delta \equiv \frac{1}{\sqrt{g}} \sum_{i,j=1}^{2} \frac{\partial}{\partial x^i} \left(\sqrt{g}\, g^{ij} \frac{\partial}{\partial x^j}\right) \tag{6}$$

with

$$(x^1, x^2) = (u,v)$$

The generalization of Thompson's two-dimensional grid equations is a direct consequence of the previous result. We know that the grid equations are given by the transformation of the following system of Poisson equations:

$$\Delta(\vec{S}) = \vec{F} \tag{7}$$

where

$$\vec{F} = (Q,R)^T$$

and

$$\vec{S} = (\eta,\tau)^T \tag{8}$$

are two twice-differentiable single valued functions. These functions define a one-to-one mapping of the domain U of the parametrization onto a rectangle. Then the Jacobian matrix, M, its inverse, M^{-1}, and the Jacobian determinant, J, exist and are given as follows:

$$M = (\vec{r}_\eta, \vec{r}_\tau) \tag{9}$$

$$M^{-1} = (\vec{S}_u, \vec{S}_v) \tag{10}$$

$$J = u_\eta v_\tau - v_\eta u_\tau$$

where

$$\vec{r} = (u,v)^T \tag{11}$$

The Laplacian operator (6) of the vector-valued function (8) is explicitly written in the following manner:

$$\Delta(\vec{S}) = \sum_{i,j=1}^{2} g^{ij} \frac{\partial^2 \vec{S}}{\partial x^i \, \partial x^j} + \sum_{j=1}^{2} \frac{\partial \vec{S}}{\partial x^j} \Delta(x^j) \tag{12}$$

with

$$\Delta(x^j) = \frac{1}{\sqrt{g}} \sum_{i=1}^{2} \frac{\partial}{\partial x^i} (\sqrt{g}\, g^{ij}) \tag{13}$$

being the Laplacian of the single valued functions $x^1 = u$ and $x^2 = v$. It follows from equations (12) and (10) that the system of Poisson equations (7) are simply

$$M_u^{-1} \begin{pmatrix} g^{11} \\ g^{21} \end{pmatrix} + M_v^{-1} \begin{pmatrix} g^{12} \\ g^{22} \end{pmatrix} + M^{-1}\Delta(\vec{r}) = \vec{F} \tag{14}$$

Now if we multiply the system of Poisson equations (14) by the Jacobian matrix (9) we get in a straightforward fashion the generalized two-dimensional elliptic grid equations

$$E(\vec{r}) = J^2 \Delta(\vec{r}) \tag{15}$$

where

$$E(\vec{r}) = \alpha\, \vec{r}_{\eta\eta} - 2\beta\, \vec{r}_{\eta\tau} + \gamma\, \vec{r}_{\tau\tau} + J^2 Q\, \vec{r}_\eta + J^2 R\, \vec{r}_\tau$$

$$\Delta(\vec{r}) = \frac{1}{\sqrt{g}} \sum_{i,j=1}^{2} \frac{\partial}{\partial x^i} \left(\sqrt{g}\, g^{ij} \frac{\partial \vec{r}}{\partial x^j}\right) \quad \text{(see (13))}$$

and

$$\begin{aligned} \alpha &= g^{11} v_\tau^2 + g^{22} u_\tau^2 - 2g^{12} u_\tau v_\tau \\ \gamma &= g^{11} v_\eta^2 + g^{22} u_\eta^2 - 2g^{12} u_\eta v_\eta \\ \beta &= g^{11} v_\eta v_\tau + g^{22} u_\eta u_\tau - g^{12} (u_\eta v_\tau + v_\eta u_\tau) \end{aligned} \tag{16}$$

A two-dimensional grid is obtained numerically, from the grid equations (15), as the solution of a Dirichlet problem where the values of the coordinates are specified on the boundaries of the parameter's domain U(1). Since the coordinates lie in U, the physical coordinates of the surface-fitted grid are given by the parametric equations (1). This is summarized in Fig. 1, where the domain V is the transformed rectangle.

THOMAS AND THOMPSON GRID EQUATIONS

Examination of the generalized two-dimensional grid equations (15) shows the presence of forcing terms. These are linked to the curvature of the surface and essentially input this information into this system. If the surface is planar, $z = 0$, with the canonical parametric equations

$$\begin{aligned} x &= u \\ y &= v \end{aligned} \tag{17}$$

The metric tensor (3) is given by the identity matrix. Hence the forcing terms vanish and the present method yields the original equations proposed by Thompson as expected.

The grid equations of Thomas are obtained as a mathematical restriction of Mastin and Thompson's three-dimensional grid equations. Hence his approach to the problem is clearly different from the proposed approach. However if we input his particular choice of parametric equations

$$\begin{aligned} x &= u \\ y &= v \\ z &= z(u,v) \end{aligned} \tag{18}$$

and source terms

$$\begin{aligned} Q &= \bar{Q}|\nabla\eta|^2 \\ R &= \bar{R}|\nabla\tau|^2 \end{aligned} \tag{19}$$

DISCUSSION AND RESULTS

The two-dimensional grid equations of Thomas and Thompson can be thought of as particular cases of the present method. However its limitation is given by the parametric equations. In many cases these equations might not fulfill all the requirements necessary to be useful (class $C^{2,2}$,...). Moreover the surfaces in practical cases will be defined by a finite number of points. Fortunately the present method can be used successfully with a class $C^{2,2}$ parametric patch technique (ref. 8). To be specific consider the following set of points:

$$S = \{\vec{r}_{ij} = (x_{ij}, y_{ij}, z_{ij})^T \,|\, 1 \leq i \leq M \;,\; 1 \leq j \leq N\} \tag{21}$$

where

$$\begin{aligned} M &= \text{number of columns} \\ N &= \text{number of rows} \end{aligned}$$

Then the surface can be seen as the union of $M\text{-}1 \times N\text{-}1$ patches,

$$S_{ij} = \{\vec{r}_{ij}, \vec{r}_{i+1\,j}, \vec{r}_{ij+1}, \vec{r}_{i+1\,j+1}\} \tag{22}$$

Hence for each patch it is possible to give three parametric equations (one for eac physical coordinate) of class $C^{2,2}$ which define the surface without ambiguity. The equations can be cast as follows:

$$x_{ij}^{k}(u,v) = \sum_{i,j=1}^{6} P_{ij}^{k}\, u^{6-i}\, v^{6-j} \tag{23}$$

into the present generalized grid equations (15), the results of Thomas are recovere

$$\alpha\, \vec{r}_{\eta\eta} - 2\beta\, \vec{r}_{\eta\tau} + \gamma\, \vec{r}_{\tau\tau} + \alpha\, \bar{Q}\, \vec{r}_{\eta} + \gamma\, \bar{R}\, \vec{r}_{\tau} = J^2 (g\, \Delta(\vec{r})) \tag{20}$$

where

$$\vec{r} = (x,y)^T$$

$$g\, \Delta(\vec{r}) = -G(z_x, z_y)^T$$

and

$$G = [(1 + z_y^2)\, z_{xx} - 2z_x z_y z_{xy} + (1 + z_x^2)\, z_{yy}]/(1 + z_x^2 + z_y^2)$$

$$\alpha = x_\tau^2 + y_\tau^2 + z_\tau^2$$

$$\gamma = x_\eta^2 + y_\eta^2 + z_\eta^2$$

$$\beta = x_\tau x_\eta + y_\tau y_\eta + z_\tau z_\eta$$

$$J = x_\eta y_\tau - y_\eta x_\tau$$

with

$$(x,y,z) = (x^1, x^2, x^3)$$

Finally the parametric equations (23) are used to calculate the metric tensor (3) and the forcing terms necessary in the numerical solution of the generalized two-dimensional grid equations (15).

The present method was programmed and applied to several examples. Figures 2 and 3 show such a surface-fitted coordinate system generated on a hyperbolic paraboloid and an elliptic paraboloid. Figure 4 illustrates a grid on the surface of a cylinder pierced by a Joukowski profile.

REFERENCES

1. Thompson, J.F.: Numerical Solution of Flow Problems using Body-Fitted Coodinate Systems. Lecture Series 1978-4, Computational Fluid Dynamics, March 13-17, 1978.

2. Mastin, C.S.; and Thompson, J.F.: Transformation of Three-Dimensional Regions onto Rectangular Regions by Elliptic Systems. Numerical Mathematics, Vol. 29, no. 4, pp. 397-407, 1978.

3. Thames, F.C.: Numerical Generation of Three-Dimensional Body-Fitted Curvilinear Coordinate Systems for Fluid Dynamics Calculations. Open Forum Paper, AIAA, 3rd Computational Fluid Dynamics Conference, Albuquerque, N.Mex., 1977.

4. Camarero, R.; and Reggio, M.: Three-Dimensional Body-Fitted Coordinates for Turbomachine Applications. ASME, WAM, Nov. 15-20, 1981.

5. Hindman, R.G.: Generalized Coordinate Forms of Governing Fluid Equations and Associated Geometrically Induced Errors. AIAA Journal, pp. 1359-1367, Oct. 1982.

6. Thomas, P.D.: Composite Three-Dimensional Grids Generated by Elliptic System. AIAA Journal, Vol. 20, pp. 1195-1202, 1982.

7. Bass, J.: Cours de Mathématiques. Masson et Cie, Tome I, Fascicule 2, pp. 30-77, 5e édition, 1978.

8. Rogers, D.F.; and Adams, J.A.: Mathematical Elements for Computer Graphics. McGraw-Hill Book Company, pp. 157-185, 1976.

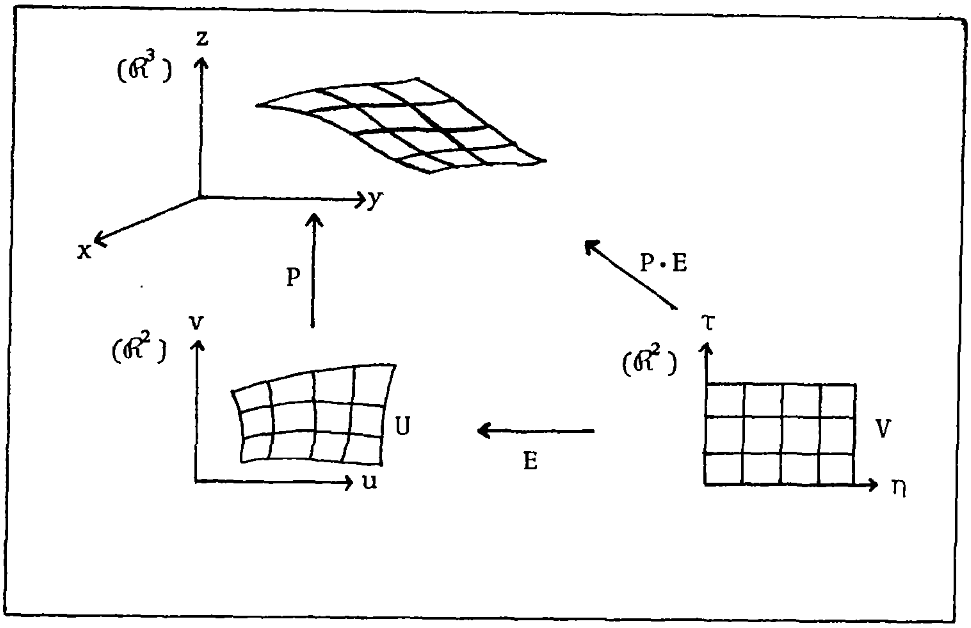


Figure 1. Transformation of surfaces.

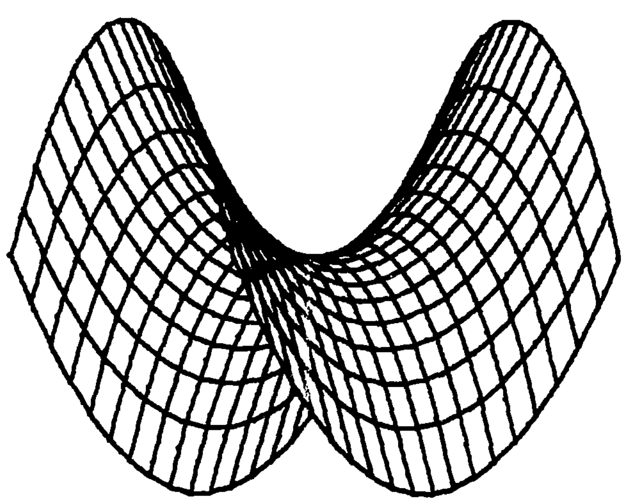


Figure 2. A surface-fitted coordinate system on a hyperbolic paraboloid.

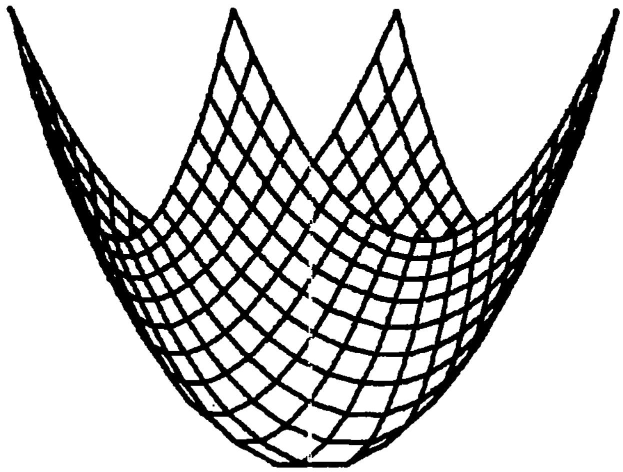


Figure 3. A surface-fitted coordinate system on an elliptic paraboloid.

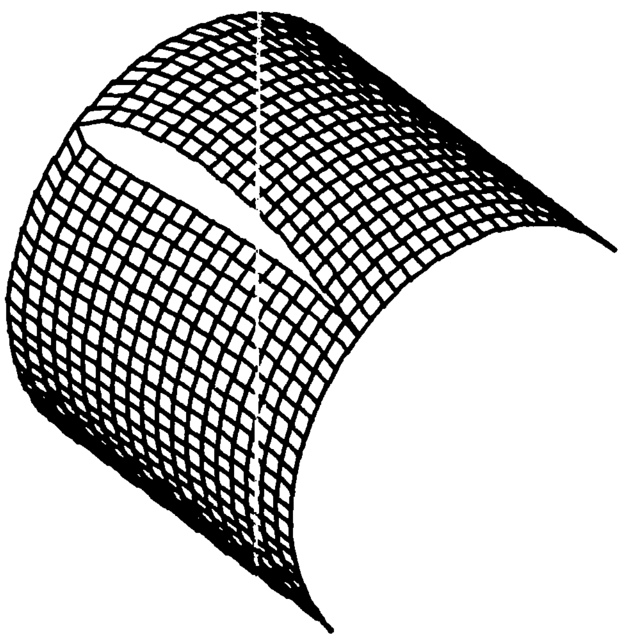


Figure 4. A surface-fitted coordinate system on the surface of a cylinder pierced by a Joukowski profile.

APPLICATION AND ENHANCEMENTS OF MOVIE.BYU

Raymond Gates and William Von Ofenheim
Computer Sciences Corporation[1]
Hampton, Virginia

MOVIE.BYU (MOVIE.BRIGHAM YOUNG UNIVERSITY) is a system of programs for the display and manipulation of data representing mathematical, architectural, and topological models in which the geometry may be described in terms of panel (n-sided polygons) and solid elements or contour lines.

The MOVIE.BYU system has been used in a series of applications at LaRC. One application has been the display, creation, and manipulation of finite element models in aeronautic/aerospace research. A typical model is shown in figures 1, 2, and 3 in line drawing, flat-shaded, and smooth-shaded renderings. These models have been displayed on both vector and color raster devices, and the user has the option to modify color and shading parameters on these color raster devices. Another application involves the display of scalar functions (temperature, pressure, etc.) over the surface of a given model. This capability gives the researcher added flexibility in the analysis of the model and its accompanying data. Limited animation (frame-by-frame creation) has been another application of MOVIE.BYU in the modeling of kinematic processes in antenna structures.

Several enhancements have been made to the MOVIE.BYU package at LaRC. The capability to handle models containing up to 4000 nodes and elements (up from 250 nodes/elements in the original version) has been added to the system. Another enhancement allows the user to input command sequences to the DISPLAY program of MOVIE.BYU from a disk file. This command input may be accomplished by either using a command file stored from a previous MOVIE.BYU session or by creating a file of MOVIE.BYU commands using the PRIME editor. An additional capability allows the storage (at the user's discretion) of an image on an off-line file for later processing.

A graphics postprocessor has been developed at LaRC to interface MOVIE.BYU to the AED-512 color raster terminal, the PRINTRONIX dot-matrix printer, the DICOMED film writer, and the Gould DeAnza IP8500 image processing system.

MOVIE.BYU is a system of seven interactive, user-friendly programs written in FORTRAN-77, residing on a PRIME 750 computer running PRIMOS 18.2. Each program (with the exception of UPDATE, which has not been implemented) has been linked to the AVID (Aerospace Vehicle Interactive Design) executive so that all of the programs can be accessed through a menu.

---

[1]Work performed under Contract Number NAS1-16078.

The seven programs that make up the MOVIE.BYU system and a brief description of each program's capabilities are listed below.

- DISPLAY - program to display triangular and quadrilateral elements
- UTILITY - program for data editing and generation
- TITLE - program that generates two- and three-dimensional characters
- SECTION - program that allows the user to process solid element data so that the data is compatible with the DISPLAY program. The program also allows the user to define clipping planes in order to modify a given model
- MOSAIC - program that converts contour line definitions into polygonal element mosaics for DISPLAY
- COMPOSE - program that allows the user to create multiple-image line drawings
- UPDATE - program used to convert old (pre-1979) MOVIE.BYU-format files to new format

All programs listed above interface to TEXTRONIX 401x-series terminals. The DISPLAY program has additional interfaces to the AED-512 color terminal and can write an image to an off-line file (disk or tape). The LaRC-implemented postprocessor allows the user to display a stored image on the following devices: the AED-512 color terminal, the PRINTRONIX dot-matrix printer, the DICOMED film writer, and the DeAnza IP8500 image processing system.

Documents that describe the MOVIE.BYU system and its implementation at LaRC are found in reference 1 and in the NASA-LaRC Implementation of MOVIE.BYU notes available from this author.

In the coming year, additional interfaces to new graphics software and hardware will be examined and implemented. In order to enhance the capabilities of the system, interfaces to existing analysis packages (i.e. SAP, SPAR, ANVIL4000, CDS, PATRAN, etc.) will be developed. Additionally, the capability to merge multiple raster images into a single raster image will be examined.

REFERENCES

1. Christiansen, H.; and Stephenson, M.: MOVIE.BYU Training Manual. Brigham Young University, Department of Civil Engineering, 1982.

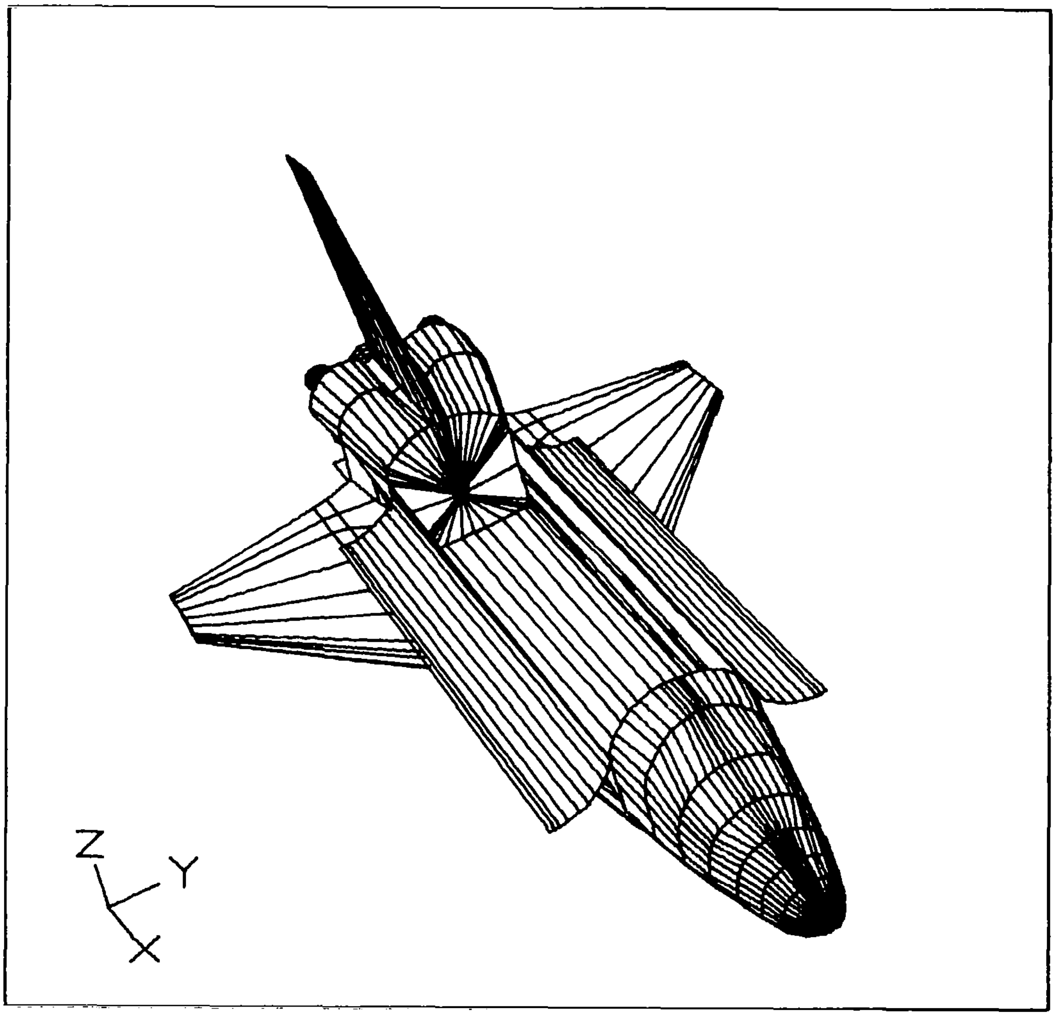


Figure 1

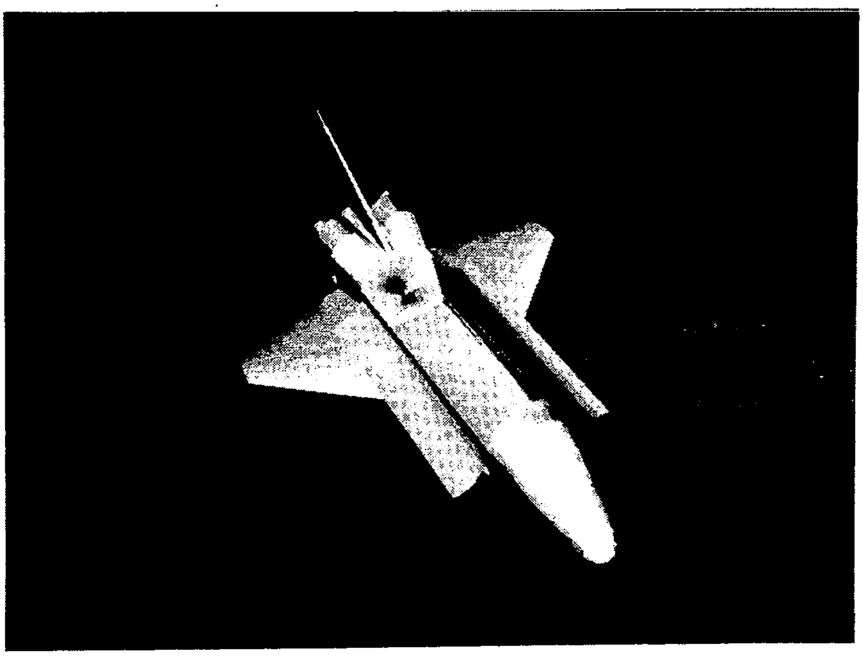


Figure 2

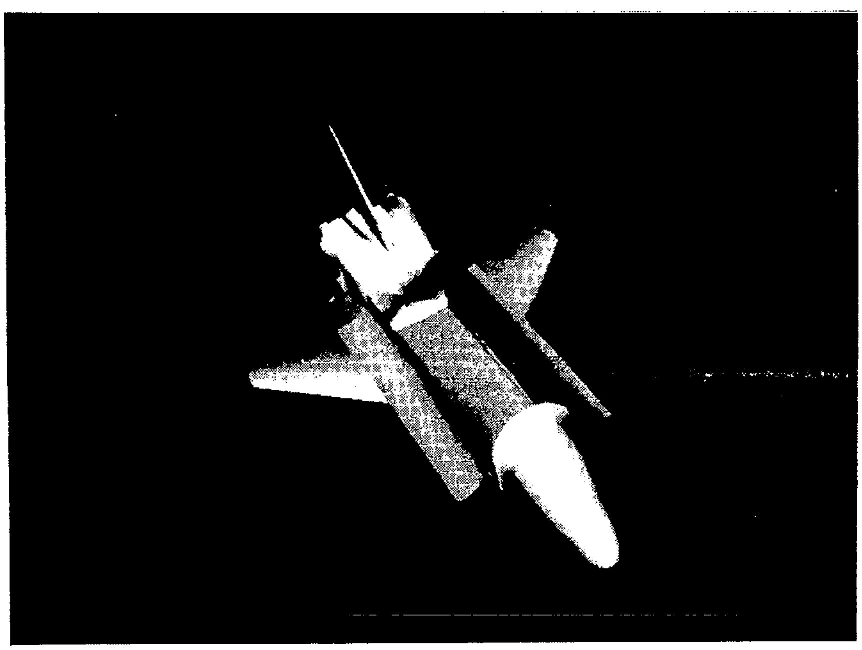


Figure 3

DESKTOP COMPUTER GRAPHICS FOR RMS/PAYLOAD HANDLING FLIGHT DESIGN

D. J. Homan
NASA Johnson Space Center
Houston, Texas

In planning Space Transportation (STS) flights, four major areas lend themselves to computer graphics modeling; these areas are payload manifestation, remote manipulator system (RMS) operations, crew visibility, and flight checklist and cue-card graphics. To this end, a graphics modeling program has been written for a desktop computer system (the Hewlett-Packard 9845/C).

The desktop system was chosen because it is easily accessible to the average engineer who has little knowledge of computers and complex computer graphics but who needs accurate pictorial representations of many possible physical object combinations in a reasonably short period of time, i.e., an electronic back-of-the-envelope solution.

The program, MADRAS (Multi-Adaptive Drawings, Renderings, and Similitudes), is written in BASIC, uses modular construction of objects, and generates both wire-frame and hidden-line drawings from any viewpoint. Objects are constructed from a menu of basic shapes (boxes, cylinders, flat plates, and surfaces-of-revolution). The dimensions and placement of these objects are user definable. Once the hidden-line calculations for a particular viewpoint are made, the viewpoint may be rotated in pan, tilt, and roll without further hidden-line calculations. The use and results of this program, as applied to the four areas of STS planning previously mentioned, are discussed in this paper.

The Space Shuttle orbiter has a 15-ft x 15-ft cargo bay area for carrying aloft numerous payloads of various sizes, shapes, and weights. As part of its complement of standard equipment, it also carries a 50-ft mechanical arm for deploying and retrieving these payloads.

In manifesting payloads for STS flights, weight and center of gravity (c.g.) are of major concern to orbiter performance; while physical clearance, accessibility, and visibility are imperative to the RMS operations. Locating payloads in the cargo bay to meet weight and c.g. constraints is more or less a standard bookkeeping task. When payloads are to be deployed or maneuvered with the RMS, however, physical clearances among the orbiter, payloads, and RMS become extremely important and are a prime consideration during the manifestation process prior to each flight. MADRAS is used in this vein to build up payload geometries and position them in the orbiter payload bay to verify adequate clearances for all RMS operations (fig. 1).

Many payloads require the RMS for more than just deployment. Some, as in the case of the Plasma Diagnostic Package (PDP) flown on STS-3, use the RMS to position them at various positions and attitudes in and around the orbiter and payload bay. Figure 2 depicts the deployment of the Long Duration Exposure Facility (LDEF) and a standard automatic trajectory that also uses the LDEF. In these cases, MADRAS is used to design the RMS trajectories that move the payload from one point to another ensuring that there are no collisions with other structures during transit The program is also used to guarantee that instrument point requirements are satisfied.

In the area of crew visibility, the RMS employs up to six closed-circuit television cameras in the performance of its tasks. The locations of these cameras are shown in figure 3. Two cameras are attached to the arm itself. One camera is located on the lower arm boom just below the elbow joint; the second is located at the wrist on top of the end effector. The elbow camera is equipped with pan, tilt, and zoom capability and gives the operator an overall view of the cargo bay and payloads during RMS operations. The end-effector camera is used primarily for the final phases of the track and capture manuever prior to grappling a payload, but it can also be used to survey various portions of the orbiter and payloads not accessibl from other cameras located in the Shuttle's cargo bay area.

There are four such CCTV cameras located in the cargo bay; two are on the forward bulkhead and two are on the aft bulkhead. These cameras are also equipped with pan, tilt, and zoom capability and are used for observing operations in and around the cargo bay. The pan and tilt option provided by MADRAS allows for knowledg of optimum positioning of these cameras for specific tasks prior to flight. Figure 4 shows the payload bay (as seen from the elbow camera and the forward bulkhead camera and a view of the grapple fixture target as seen through the end-effector camera (fig. 5).

For each phase of a flight, the crew carries a detailed checklist that outline the activities to occur at any particular time. On all previous flights and all upcoming flights on which the RMS will be used, the flight crews have requested that graphics of orbiter/RMS/payload configurations be included in their flight checklists. These provide the crews with visual definitions of configurations that would otherwise just be tables of numbers in their checklists, thus providing them with another means of configuration verification. Figure 6 shows RMS configurations to be used on the STS-7 flight to validate ground-based simulations.

MADRAS provides support as well as finished products to these areas in a timely manner and is less cumbersome to use than large computers and complex graphics systems.

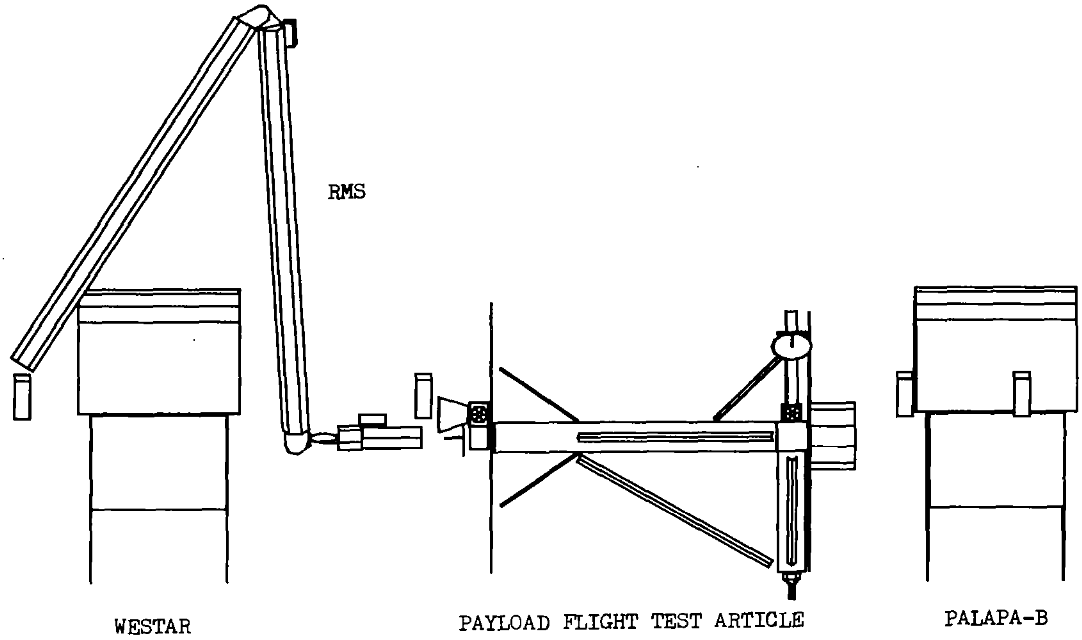


Figure 1.- STS-11 payload bay manifest.

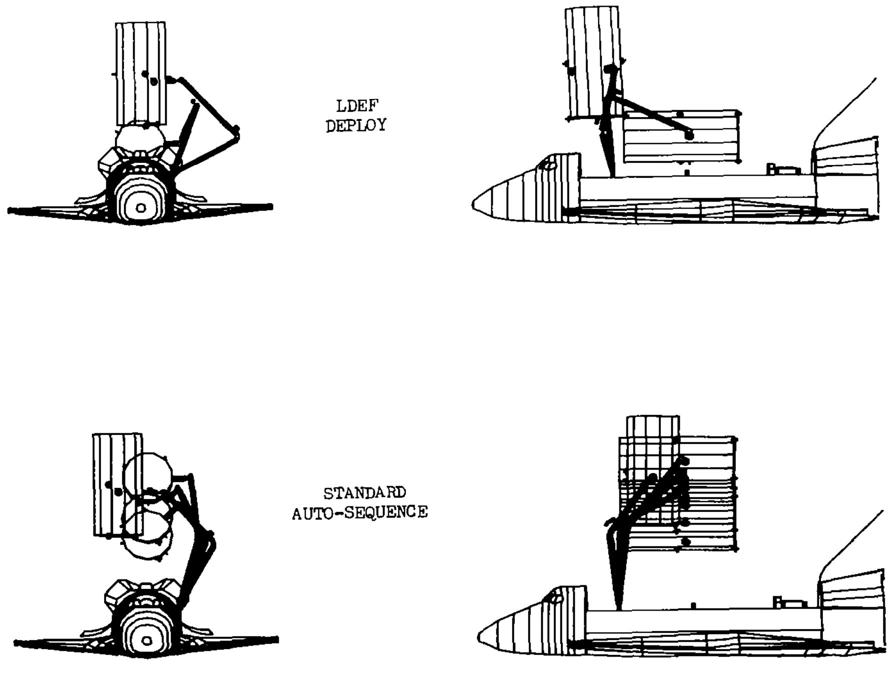


Figure 2.- LDEF deployment and standard automated sequence.

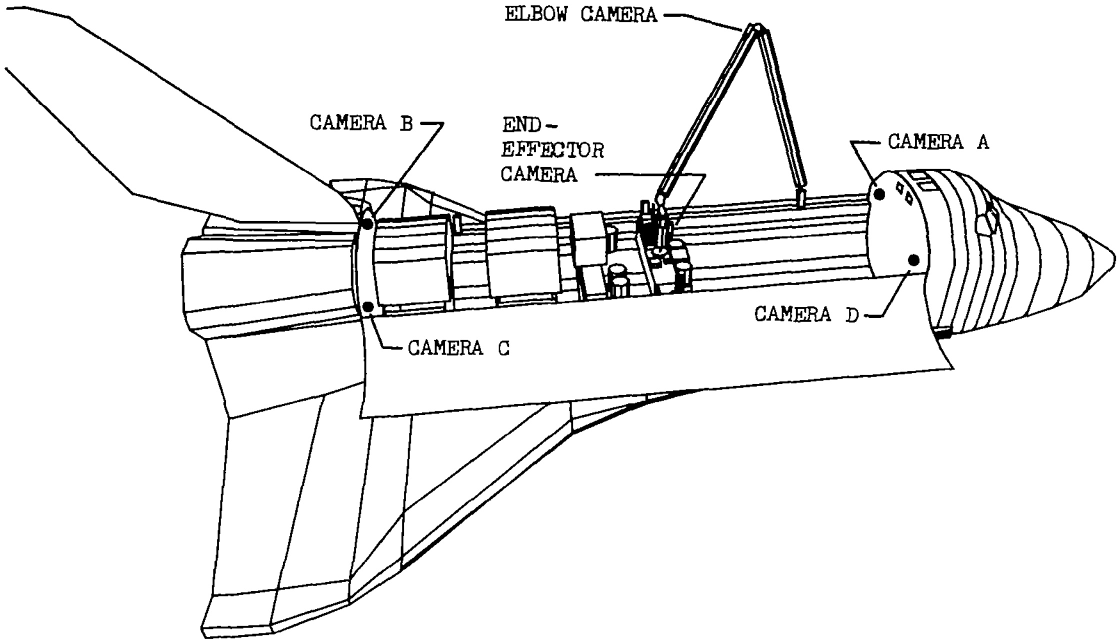


Figure 3.- Closed-circuit television camera locations.

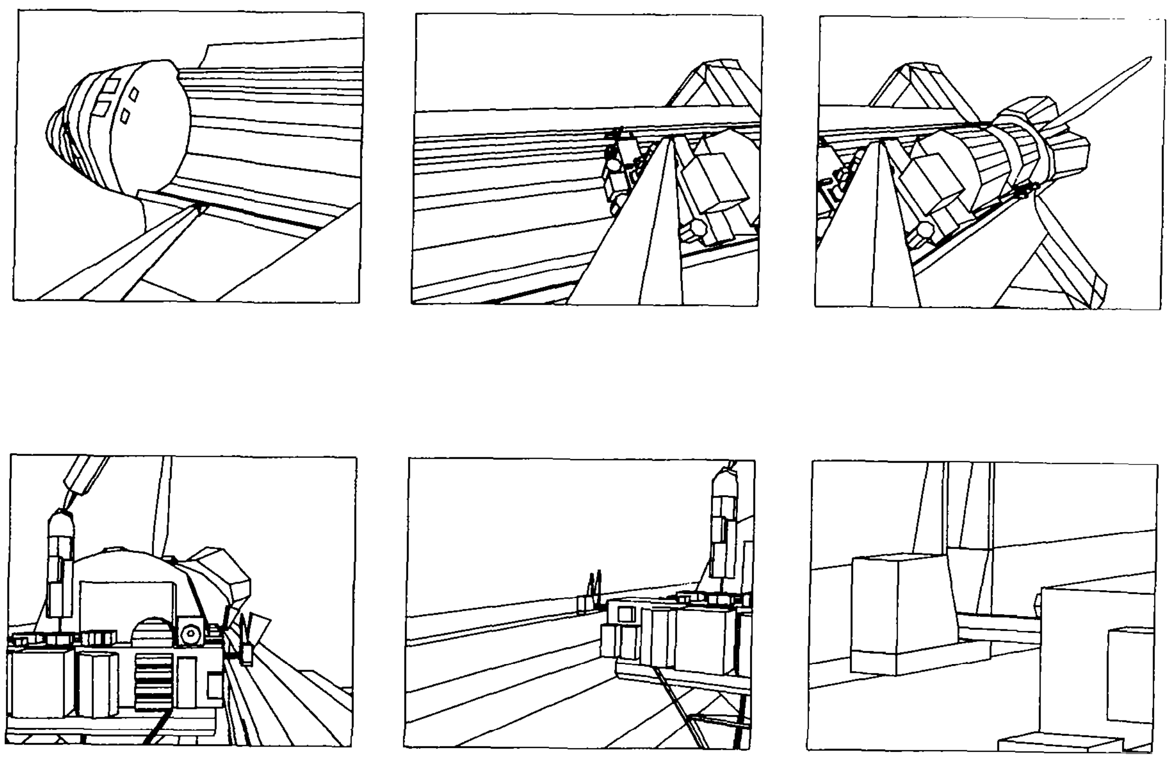


Figure 4.- STS-7 payload bay as seen from the RMS elbow camera (top) and the forward bulkhead camera A (bottom).

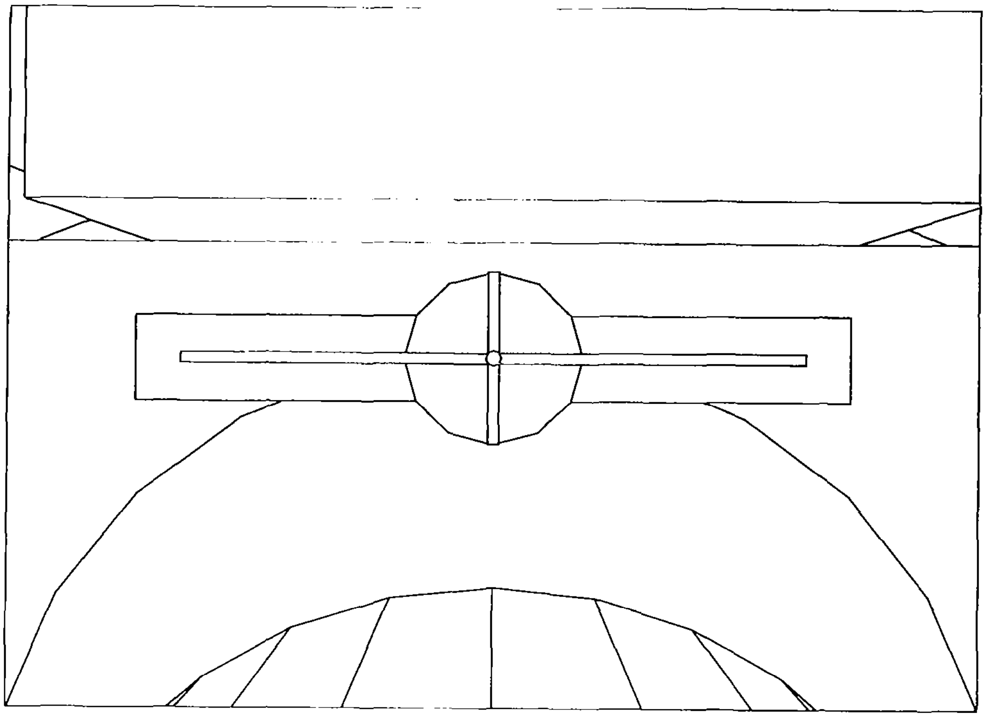


Figure 5.- View of grapple fixture target as seen through the end-effector camera.

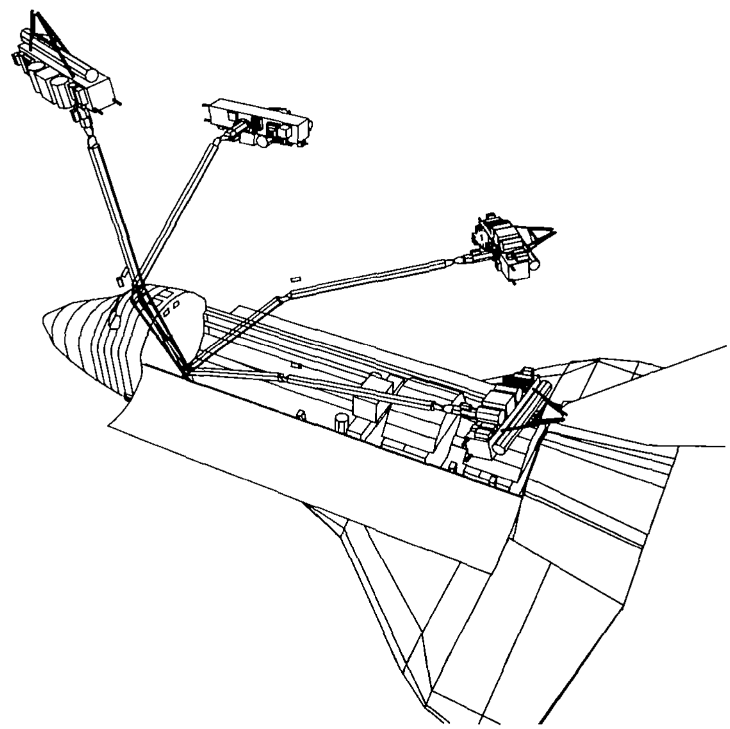


Figure 6.- RMS configurations for STS-7 common validation runs.

AUTOMATIC GENERATION OF 3-D ENVELOPES

James Leavitt
Princeton University
Princeton, New Jersey

Walter Messcher
Transportation Systems Center
Cambridge, Massachusetts

The object of this research is the automatic generation of three-dimensional envelopes. Using the computer, an envelope surface for any object moved along any path may be developed. In this way, the geometry of physical objects and regions may be modeled. The FORTRAN computer language is used, and resulting envelopes are plotted pictorially.

First, the desired object and its path must be described parametrically. For instance, if one wished to move a sphere in a straight path to create a cylindrical envelope, let X, Y, and Z be functions of the parameters s, t, and w. The object (with radius R) can be defined as follows:

$$X = R \cos(t) \cos(s) + Xc(w)$$

$$Y = R \sin(t) \cos(s) + Yc(w)$$

$$Z = R \sin(s) + Zc(w)$$

Where Xc, Yc, and Zc represent the trajectory functions for the sphere's center. For a straight line trajectory in the X direction:

$$Xc = w$$

$$Yc = 0$$

$$Zc = 0$$

From the principles of vector analysis and parametric equations, a relationship between the object and its envelope can be derived. Steps in this derivation involve crossing constant s- and t-path vectors to get an outward body normal from the object. The outward normal is then dotted with the expression for the object's path tangent. The result is a nonlinear partial differential equation relating the parametrically defined object, its described path, and the surface tangent or envelope. The resulting quantity will be called Q and is given by the following equation:

$$Q = \frac{\partial X}{\partial w}\frac{\partial Y}{\partial s}\frac{\partial Z}{\partial t} - \frac{\partial X}{\partial w}\frac{\partial Y}{\partial t}\frac{\partial Z}{\partial s} + \frac{\partial X}{\partial t}\frac{\partial Y}{\partial w}\frac{\partial Z}{\partial s}$$

$$- \frac{\partial X}{\partial s}\frac{\partial Y}{\partial w}\frac{\partial Z}{\partial t} + \frac{\partial X}{\partial s}\frac{\partial Y}{\partial t}\frac{\partial Z}{\partial w} - \frac{\partial X}{\partial t}\frac{\partial Y}{\partial s}\frac{\partial Z}{\partial w}$$

Setting Q equal to zero produces the line segments that make up the envelope surface. Resulting values of this equation may be refered to as Q values. In general, Q depends on the object's descriptive X, Y, and Z equations and involves a nonlinear relationship between paramters s, t, and w.

In order to solve the nonlinear equation, a regular three-dimensional grid is constructed in s, t, and w space. This lattice fills s, t, and w space with a series of abutting parallelepipeds. In order to further decompose this space, the parallelepipeds are divided into triangles. In each parallelepiped, thirty-six triangles are positioned to minimize empty space in a face-centered and body-centered pattern (see figure 1).

Each triangle is described spatially by its three vertices in s, t, and w coordinates. However, by invoking the Q equation numerically, each vertex can also be assigned a Q value. In order for the Q equation to produce the appropriate envelope segments, the Q value must equal zero.

The problem then becomes one of contouring with the planar triangles of the parallelepipeds. The Q values associated with each triangle vertex are compared to zero.

Three possibilities exist in this comparison. First, and least likely, the Q equal to zero value could occur exactly on a vertex, in which case that particular vertex would represent a solution point. Second, the zero value could exist between two vertices. Here, a simple linear approximation is made on the triangle segment connecting the two vertices to estimate the location of the solution point. Last, the Q equal to zero value might occur neither at a triangle vertex nor within the triangle.

In the preceding process, all s, t, and w coordinates having Q values equal to zero are stored. Then these coordinates can be connected by the line segments of the envelope surface. The process is repeated for each triangle in a parallelepiped and then for each parallelepiped in the grid. Therefore, all appropriate s, t, and w points are determined to represent the entire envelope surface. To graphically display an envelope, s, t, and w coordinates can be easily converted to X, Y, and Z points through the object's parametric equations. The X, Y, and Z points can be plotted pictorially by simple transformations that make the points two dimensional (see figures 2, 3, and 4).

Basically, the computer has been used to model the three-dimensional geometry of an object as it is moved along a path creating an envelope. Computer graphics can be used to display these paths and provide engineering applications. In robotics, it is often desirable to know or see the envelope path of a robot's movement. Also, in aviation, envelopes can be developed for flight movement. An important weakness in the methodology behind the envelope development is the regularity of the grid. The regular grid implies constant detail throughout the parallelepiped lattice so that areas that experience large Q changes can be accounted for with a highly detailed grid of parallelepipeds. However, areas that do not have large Q changes become simultaneously detailed. Therefore, by creating a very fine grid in one area, efficiency is sacrificed in the other parts of the lattice. A solution exists in the development of a variable grid in which there would be "communication" between the function Q and the grid. Research in this area is being successfully conducted at present.

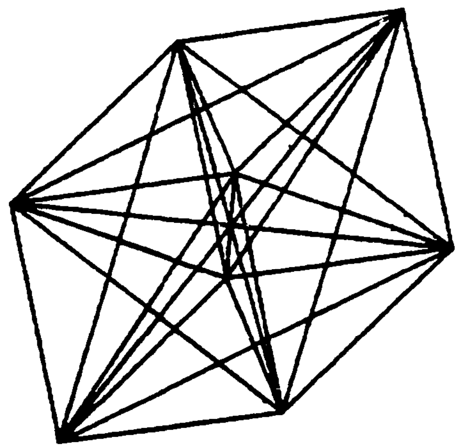


Figure 1.- Triangle scheme.

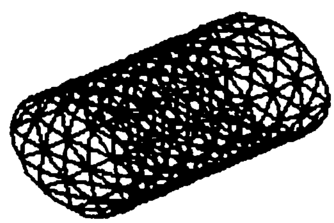


Figure 2.- Cylindrical envelope.

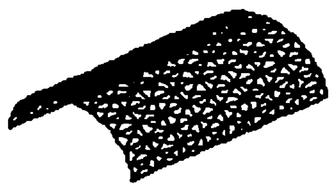


Figure 3.- Shell envelope.

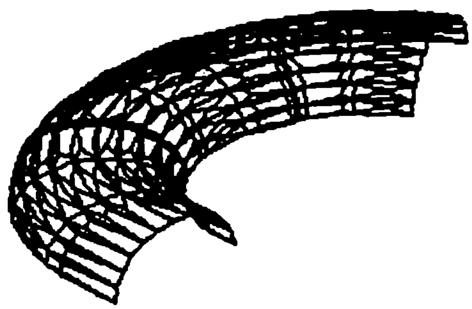


Figure 4.- Torus shell envelope.

ESCHER: AN INTERACTIVE MESH-GENERATING EDITOR FOR PREPARING FINITE-ELEMENT INPUT

W. R. Oakes, Jr.
Analysis and Testing Group
Design Engineering Division
Los Alamos National Laboratory
Los Alamos, New Mexico

INTRODUCTION

ESCHER is an interactive mesh generation and editing program designed to help the user (1) create a finite-element mesh, (2) create additional input for finite-element analysis, including initial conditions, boundary conditions, and slidelines, and (3) generate a NEUTRAL FILE that can be prostprocessed for input into several finite-element codes, including ADINA, ADINAT, DYNA, NIKE, TSAAS, and ABUQUS.

We will discuss in detail two important ESCHER capabilities, interactive geometry creation and mesh archival storage. We also describe our interactive command language and our use of interactive graphics. The archival storage and restart file is a modular, entity-based mesh data file. Modules of this file correspond to separate editing modes in the mesh editor, with data definition syntax preserved between the interactive commands and the archival storage file. Because we expect ESCHER to be highly interactive, we provide extensive user documentation in the form of an interactive HELP package.

INTERACTIVE GEOMETRY DEFINITION

We use geometry definitions from the Initial Graphics Exchange Specification (IGES), ensuring the ability to obtain original geometry from a variety of CAD/CAM systems. We defined classes of meshing entities that a user would be familiar with, such as GEOMETRY, MESH CONTROL, PARTS, and INITIAL and BOUNDARY CONDITIONS. A Mesh Definition File (MDF) stores the mesh information, and an editing language is used to create and modify this file. The mesh editing language looks like a familiar text-editing language. Interactive graphical input is an option.

MESH DEFINITION FILE

The MDF is a modular file that contains independent logical blocks of mesh entity data. We found that users wanted a mesh generator that was both easy to use and had great flexibility (localized mesh refinement, user control of interior mesh shape, etc.). These requirements are generally in conflict. To solve this problem, we created a detailed input file (MDF) that gives user control over all phases of the mesh. We also created, as part of ESCHER, an interactive editor with many default and automatic features to create the detailed MDF. Using this philosophy, the user who is interested in an easy-to-use generator can use the default values and the automatic functions to create a mesh. The user who needs more control for such things as mesh refinement can first perform the easy tasks, and then use the editor features to modify the MDF. The user can also directly create an MDF using any conventional editor in the same fashion as in conventional mesh generation.

We have defined the following blocks: GEOMETRY, MESH, PARTS, BOUNDARY, INITIAL, and SLIDELINE. GEOMETRY contains all of the line definitions and reference coordinate systems, and hence the spatial (X, Y, Z) data. MESH contains the number of nodes and node distribution information for each line. PARTS contains data that define how the lines are connected to construct parts and what kind of elements will be generated for the parts. BOUNDARY, INITIAL, and SLIDELINE blocks each contain lists of lines to which the user has assigned various boundary conditions and other specifications. The user may use the editor to generate and modify some or all of the blocks. Finally, the user will generate both an updated MDF and a NEUTRAL FILE. The NEUTRAL FILE is compatible with the rest of our finite-element analysis system. ESCHER can convert a complete MDF to an analysis input file with no user interaction. The MDF is a much more compact file than the NEUTRAL FILE (the NEUTRAL FILE is equivalent to a node-by-node, element-by-element, finite-element input deck). Consequently the MDF is a good candidate for archival storage.

INTERACTIVE COMMANDS

ESCHER interactive commands perform editing functions on the meshing entities (geometry, mesh control, parts, boundary conditions, initial conditions, and slidelines). Editing functions for each entity include ADD, COPY, DELETE, REPLACE, and UPDATE. Commands to TYPE or PLOT data associated with these entities are also available. In addition, there are commands to find line intersections and to create parts automatically. We used text editor syntax as a model for the interactive command syntax. Key words need to be specified only to uniqueness, usually two letters. Examples of each command are available through the HELP command.

INTERNAL DOCUMENTATION (HELP PACKAGE)

In the modern interactive computing environment, users normally have questions while they are sitting at their terminal. Therefore, we use internal documentation rather than an external user's guide. The documentation format we chose is a multilevel help package. Although our documentation is more detailed than normal VAX/VMS system help packages, the philosophy is the same.

ESCHER is a multilevel interactive code. This means that at the highest level, the user may choose to work in one of several broad areas, such as geometry, part definition, or boundary definition. If the user types HELP at any level, ESCHER will give a description of the purpose of this level plus a list of acceptable commands. The user may then type HELP "any command" and ESCHER will type a complete description of that command, with a description of any additional parameters. The HELP package can be operated either at the VAX/VMS level, for user information, or during the execution of ESCHER.

MESH ALGORITHMS

We chose a mapping method to implement our meshing algorithms. This technique is compatible with the geometry specification formats (IGES) available to us for the next five to ten years. The IGES format contains a good geometry interface standard that will be supported by virtually all CAD/CAM systems with which we will interface for the foreseeable future. Our implementation of the mapping method uses the linear blending functions developed by Cook for INGEN [1,2]. To allow great flexibility in mesh definition and grading, we use a three-step mesh generation technique. First, we create the mesh in a square (or cubical) mapping space. Second, we relax the mesh in the mapping space. Finally, we map the mesh into the model space. In the final process, a two-step mapping assures element compatibility between parts and allows variable control node spacing along edges while maintaining geometry integrity.

CONCLUSIONS

ESCHER required the development of (1) new meshing algorithms, (2) new finite-element data file format standards, (3) new highly interactive graphical software and a new code operating philosophy, and (4) new links with CAD/CAM systems and physics design codes. ESCHER is now in a "friendly user" status in our group. We hope to be able to release it for laboratory use in the near future.

REFERENCES

1. Cook, W. A.: Body Oriented (Natural) Coordinates for Generating Three-Dimensional Meshes. Int. J. Numer. Meth. Engr., vol. 8, 1974, pp. 27-43.

2. Cook, W. A. : INGEN: A General-Purpose Mesh Generator for Finite Element Codes. Los Alamos Scientific Laboratory Report No. LA-7135-MS, March 1978.

THE SOLID MODEL GEOMETRY GENERATOR (SMGG)

By

K. H. Jones, D. P. Randall,
R. L. Gates, and W. H. Von Ofenheim
Computer Sciences Corporation*
Hampton, Virginia

The Solid Model Geometry Generator (SMGG) system provides the capacity to model complex solid objects through the composition of "primitive" parts. The parts are specified through input of dimension and orientation parameters. The parts may later be edited, the model may be displayed (through a variety of display options), and the resulting geometry formatted for subsequent analysis.

The SMGG system was originally constructed for use in the preliminary stages of space station and other spacecraft design. Because the system models a design through the use of basic shapes common to many solid objects and provides limited capabilities for producing irregularly shaped parts, it can be used for a variety of applications. (See Figures 1-4)

Currently, seven primitive parts may be specified and produced through SMGG. These include "boxes", spheres", "cones", "paraboloids", "tori", and translationally or rotationally "swept" parts. (See Figure 5) Each part may be specified by the user through input of basic part dimensions and a local orientation. For example, a box is specified by its length, width, height and orientation in space (rotation, translation, and scaling).

Although only a limited number of primitives explicitly exist, numerous other shapes can be generated. For example, a true cone is a "cone" primitive with one end radius equal to zero. A cylinder is a "cone" with both radii equal and a truncated cone is a "cone" with unequal radii neither of which is zero. Elliptical parts may be produced through the specification of one or more scaling parameters. In addition, the sphere, cone, paraboloid, and tori are specified by supplying the number of longitudinal (and/or latitudinal) facets. Varying the number of these facets will generate a new shape. Finally, for some primitives it is possible to generate a "slice" (or "wedge") by varying the revolution angle.

The translationally swept part provides the capability to produce an irregularly shaped "cylinder". The user creates a closed curve and then translates this curve in any direction to create the three dimensional part. Similarly, a curve may be specified and then rotated around the x-axis to form a general surface of revolution.

*Work performed under Contract Number NAS1-16078.

Another means for creating an irregularly shaped part is provided through the "external" part. By this means, the geometry data for a part (node coordinates and connectivity information) may be formatted on a file, read into the system, and added to the parts file in lieu of part dimensions.

Boolean operators provide the means of forming single parts that are unions, intersections, or differences of other parts. Parts may also be grouped for convenience into "assemblies". In an assembly the parts maintain separate boundary representations but can be collectively referenced, transformed, copied, or saved (to be read into another model).

For each part, the user supplies local orientation parameters for rotation, translation and scaling. Rotation and translation are used to position the part relative to other parts. In an assembly, the local orientation parameters are applied to each part and then all parts are transformed by the orientation parameters for the assembly.

The part dimensions and orientation parameters for each part are maintained in core until termination of the system. At this point the user is allowed to save this information on a "parts file" to be read by the system at a later time for reconstruction of the model.

Two editors are provided by SMGG. A text editor provides the capability to add, delete, modify and copy part dimensions and orientation parameters of parts. Subsequently, the displayed model will reflect these changes. A graphics editor provides the capability to graphically transform parts using graphical input devices. Parts may be translated or rotated without the need for knowledge of their translation or rotation parameters. Nodes and faces of an individual part may be added, deleted, or modified. Coordinate values of nodes may be obtained and dimensions between nodes or part centroids may be displayed.

The user is allowed a variety of display options. He may select the parts to be displayed (all parts, individual parts, or ranges of parts). He may provide global orientation parameters that will be applied to all displayed parts. Part labels may be displayed and edges may be shrunk (to separate faces). The model may be displayed with hidden lines or hidden surfaces removed, or flat or smooth shading. (See figures 6-12) The user may repeatedly zoom on the model or display the model in four simultaneous views (XZ, YZ, ZX, and user specified).

Because parts are described by the user in terms of basic dimensions (length, radius, number of sides, etc.) and orientation (rotation, scaling, and translation) rather than by specifying node coordinates, less input is required and fewer calculations need be made by the user. The geometry data is calculated from the description and stored in a temporary file (transparent to the user). To save a model, only the parts file (containing the description) need be stored. This provides a substantial savings in storage since a part description requires significantly fewer entries than the node and connectivity information for that part.

A more subtle but most important advantage of this method is the natural means by which parts are created and modified. Many systems require the user to describe primitives in terms of node coordinates and connectivity data. By separating a parts description from its orientation in space, the SMGG system frees the user of the need for this knowledge and allows him to describe a part in more natural terms (its dimensions) and afterward to deal with its orientation in space. Unless he desires so, he will never need to know the coordinates of the nodes or their connectivity.

SMGG is divided into subsystems whereby the features described above are implemented as subsystem commands. The entire system is menu driven with extensive prompting and on-line "help" commands. Internal error checking is provided for all user input.

In addition to creation of a parts file, a geometry file is created containing a boundary representation (node coordinates and connectivity information) for each part (similar to a finite element model). This file is normally transparent to the user and he need not be concerned with its appearance or function. However, if desired, the geometry file may be saved, edited external to the system, and read back into the system for display. The geometry file may also be converted into a format for input to the MOVIE.BYU graphics system. Because of its ease for creating and editing parts, SMGG serves as an excellent means of building a model for display with MOVIE.BYU. The geometry file may be saved and fed as input to a finite element analysis program. Data from applications programs may be used by the system if formatted as a parts file, geometry file, or external part file.

The SMGG system was written in FORTRAN 77 for a PRIME 750 using the PRIMOS 18.2 operating system. It is interfaced to the TEKTRONIX 4010 series terminals and the AED 512 color graphics terminal. Both on-line and off-line documentation are available which provide detailed information on the system features and command language.

Plans for future releases of SMGG include the ability to model interiors of solids in addition to exteriors. This would require the capability for cross sectioning ("slicing") and transparency of parts. Also, limited mass properties analysis will be provided (center of gravity, moment of inertia, etc.). Future releases will take advantage of the unique capabilities of the AED particularly with respect to graphics input.

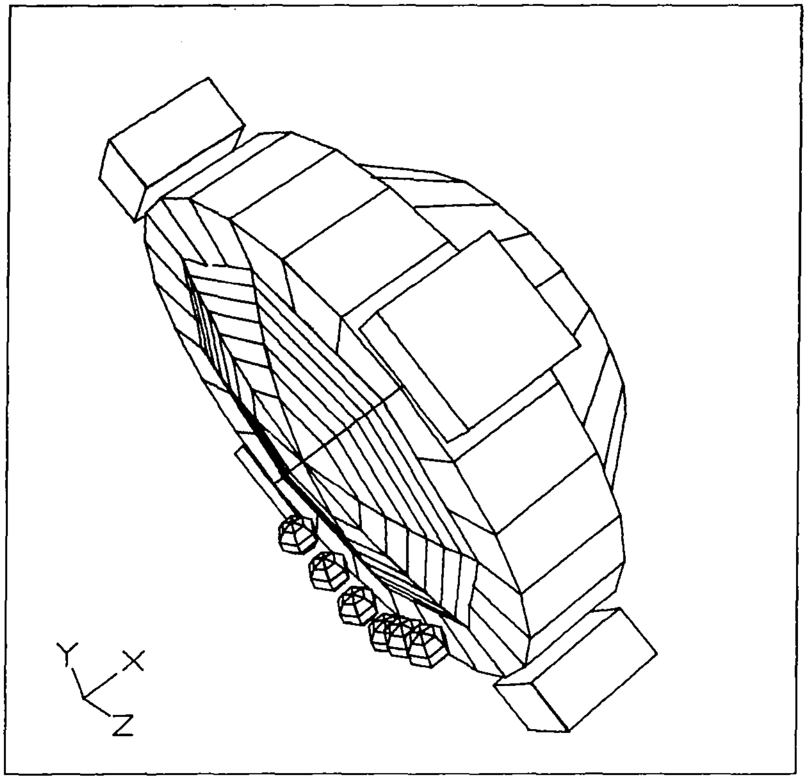


Figure 1

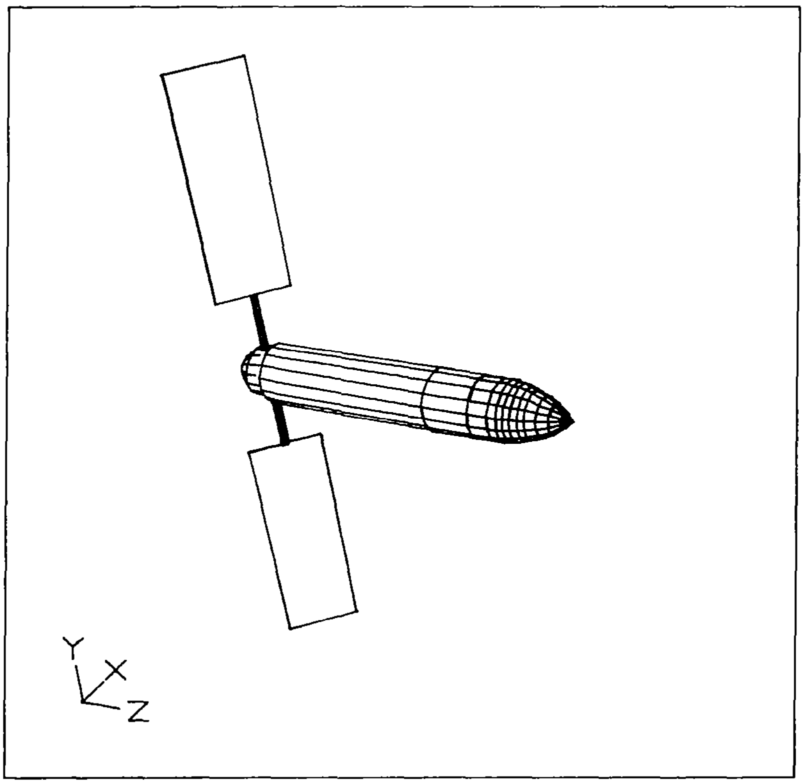


Figure 2

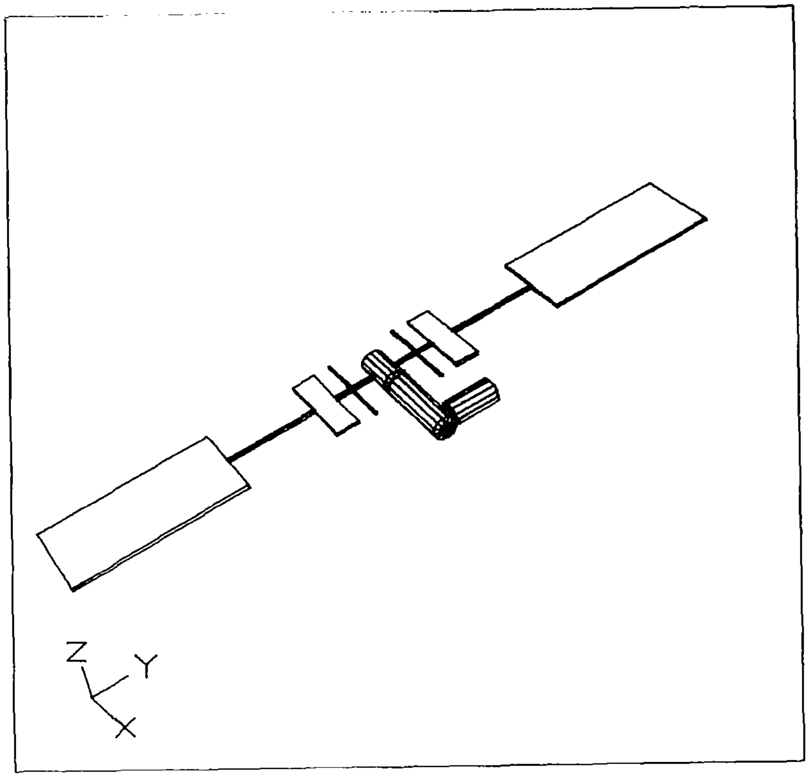


Figure 3

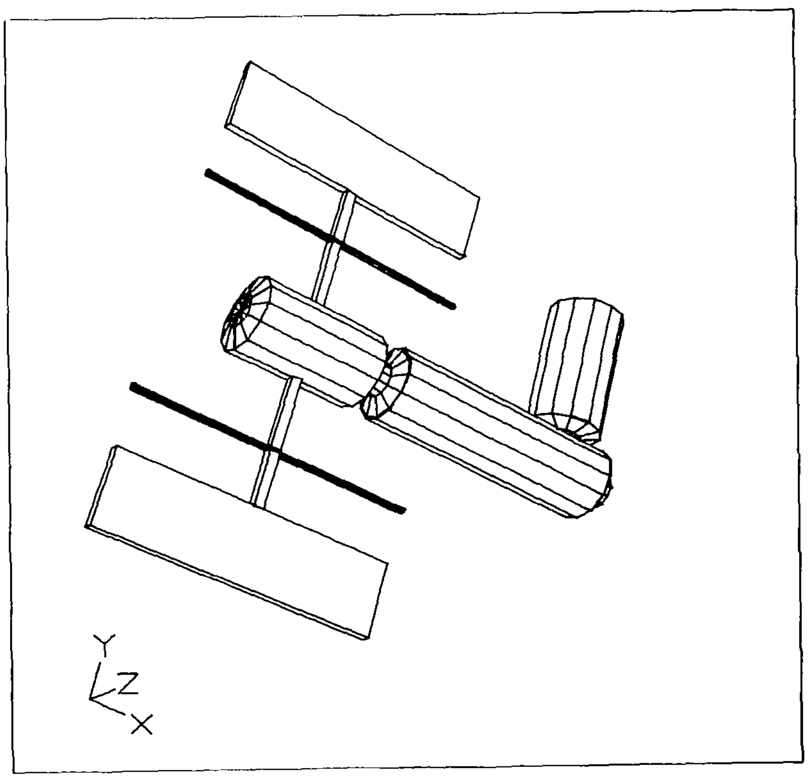


Figure 4

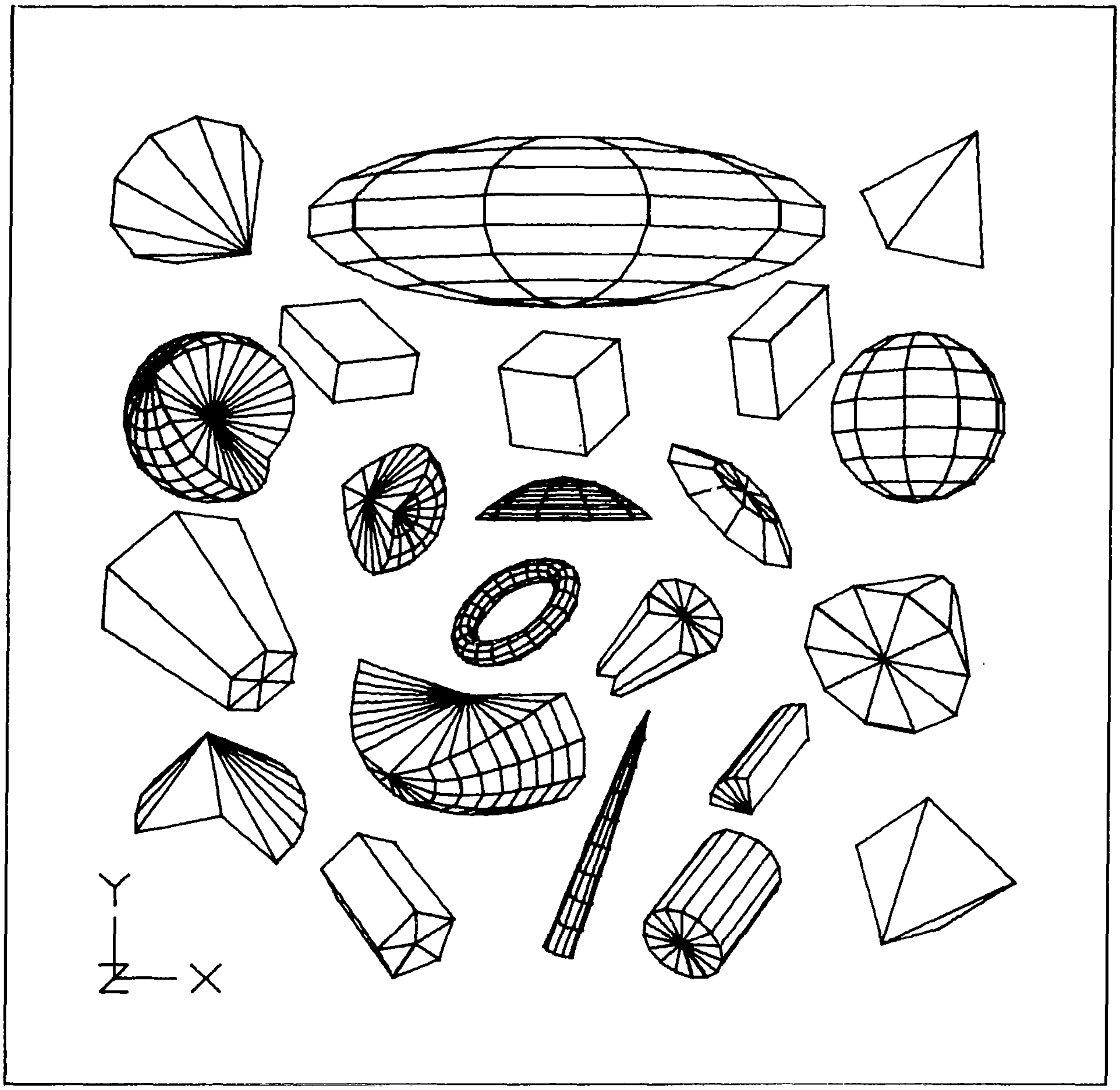


Figure 5

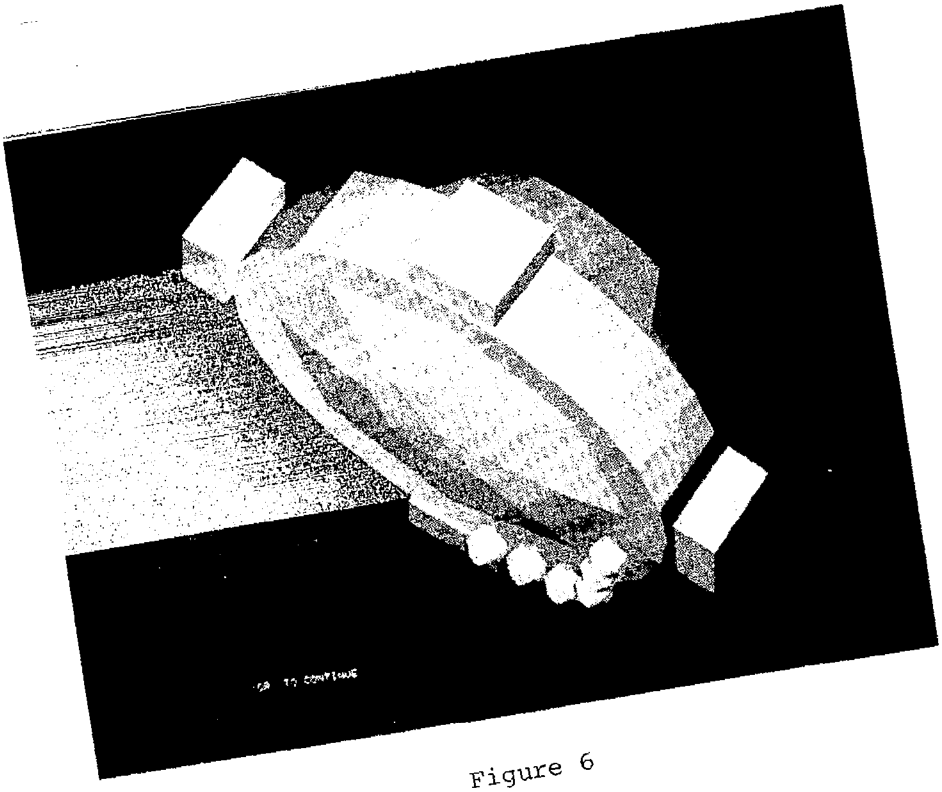


Figure 6

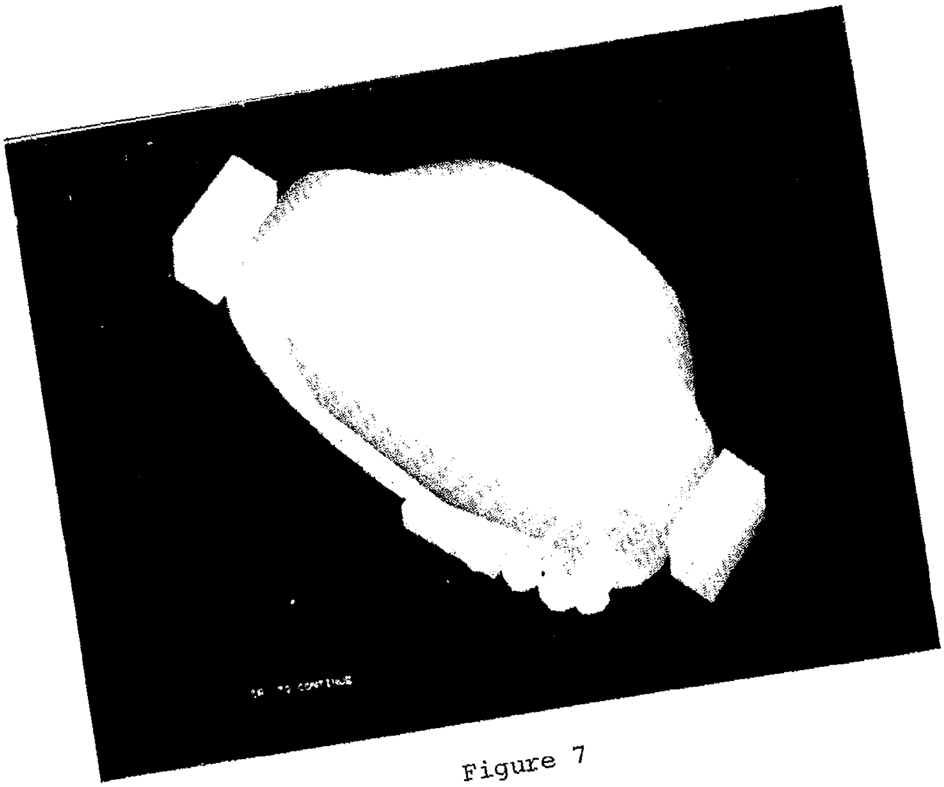


Figure 7

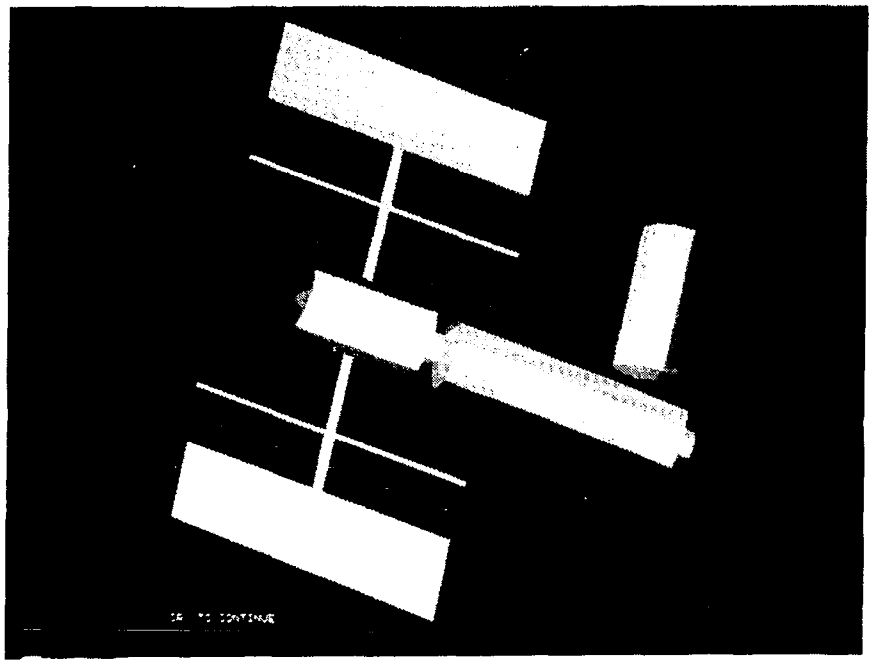


Figure 8

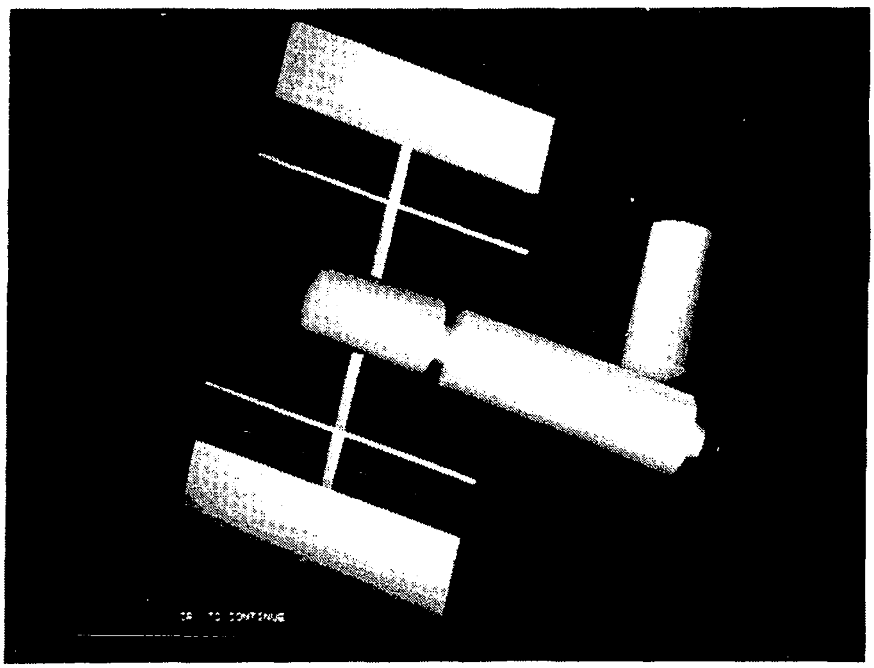


Figure 9

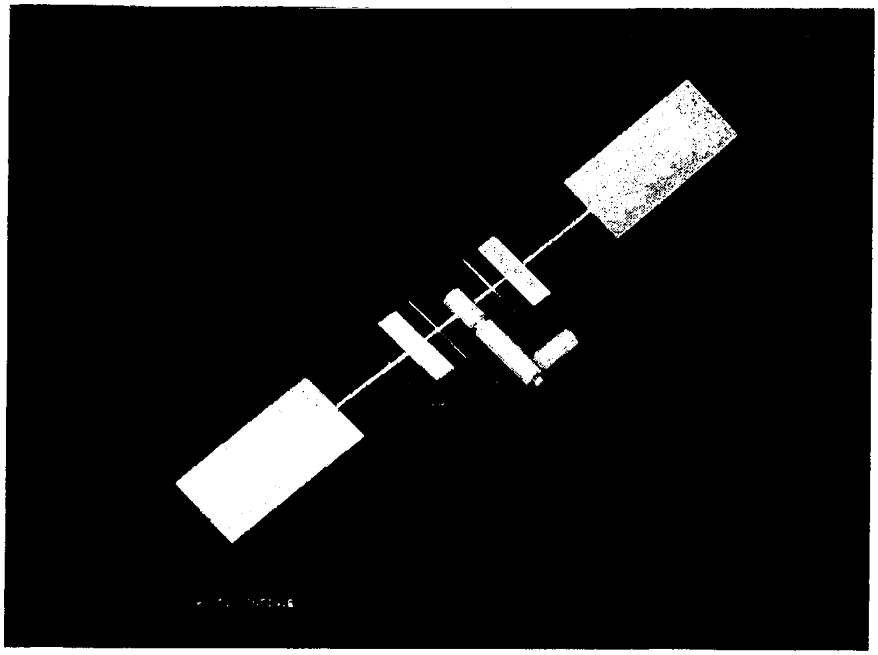


Figure 10

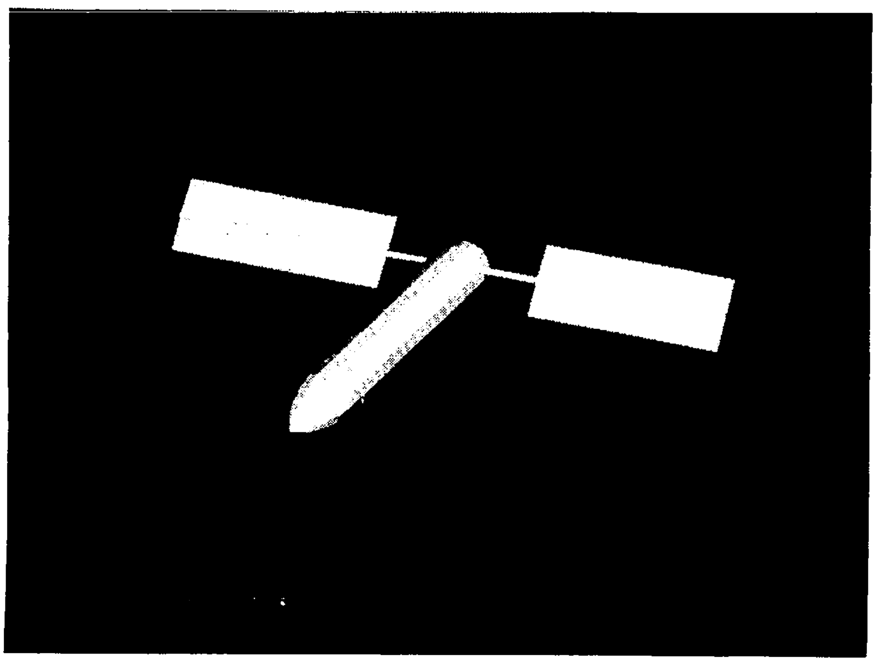


Figure 11

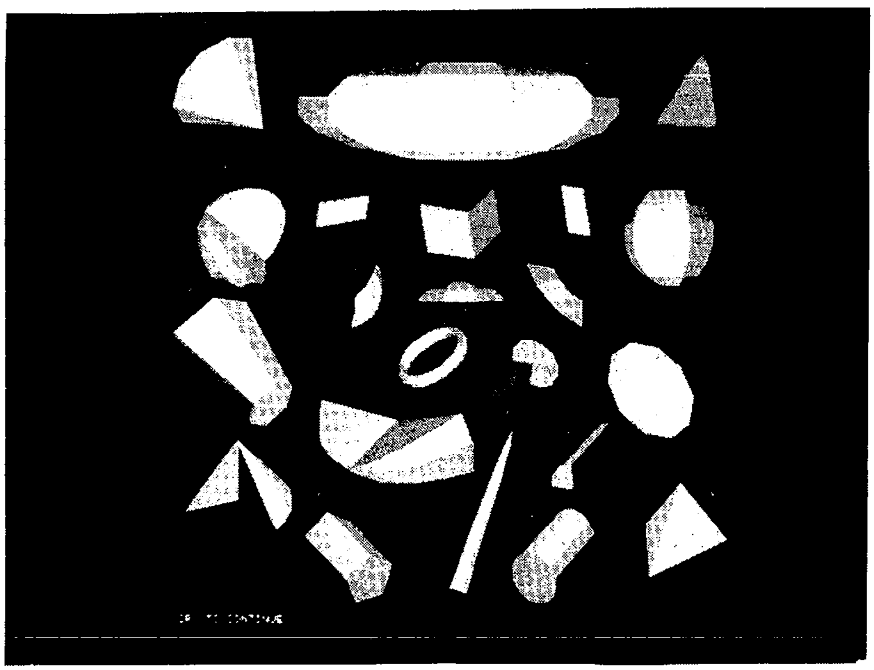


Figure 12

THE 3SPACE DIGITIZER - A NEW INPUT DEVICE FOR SOLID GEOMETRY MODELING

Jack Scully
Polhemus Navigation Sciences, Incorporated
Essex Junction, Vermont

INTRODUCTION

In recent years, solid modeling programs have been developed to enhance the design of components and the generation of object-description data bases. From his terminal, a designer can now create most simple objects through a series of keyboard commands. Nevertheless, inputting complex shapes remains a difficult and costly process, which requires significant engineering time and computer software.

Many limitations in solid modeling techniques can be overcome by a three-dimensional digitizer called the 3SPACE Digitizer, which was developed by Polhemus Navigation Sciences, Incorporated.

DERIVATION

The 3SPACE Digitizer is a derivation of SPASYN, which is a patented magnetic transducing system offering an accurate method of calculating the location of one object relative to another. SPASYN uses low-frequency magnetic fields to measure both the position and orientation of its sensor relative to its transmitter.

In the military aircraft environment for which it was originally developed, SPASYN is used to provide a continuous and precise measurement of a pilot's line of sight. In this application, a SPASYN sensor is mounted on the pilot's helmet, and a transmitter is affixed to the canopy. By looking through a reticle on his visor, the pilot can transfer the angular coordinates of a visually acquired target to a fire control system.

SYSTEM DESCRIPTION

The 3SPACE Digitizer unit shown in Figure 1 consists of a system's electronics unit, a model table, a stylus, and a keypad. In operation, the 3SPACE Digitizer determines the X, Y, Z coordinates of any point located on a low-conductive, three-dimensional model. This data, along with the orientation of the Digitizer's stylus, is immediately available for transmission to a host computer or graphics terminal. Digitization is effected by touching the point to be measured with the tip of a handheld stylus.

Physical objects or models to be digitized are first placed on a model table that provides a reference plane for measuring the model. The system is presently configured to measure the coordinates on a model up to 20 in. x 20 in. x 10 in. in size with an accuracy of 1/32 of an inch. Smaller models can be digitized with greater accuracy. Models as large as 50 in. x 50 in. x 20 in. can also be digitized but with lesser accuracy.

The stylus, which contains a sensor for measuring the transmitted field, is about the size of a pen and is shaped for digitizing concave as well as convex surfaces. Because labelling data points is important, the unit includes a keypad for instantaneous computer coding.

The system's electronics unit contains all hardware and software essential to control digitizing operations. It consists of one or more analog boards and a digital processor. The analog boards contain circuitry to generate and sense the electromagnetic fields and digitize the sensed analog signals. The digital processor, which is an 8086-based single-board computer, controls the analog boards and performs all necessary computations. The 3SPACE Digitizer communicates with the outside world through a standard RS-232C interface between itself and a user's host computer or data terminal.

Once digitized, data points can be output in either discrete or continuous modes. The output is selectable in either integer or floating point formats. Positional data is calculated in X, Y, Z coordinates, while orientation information can be determined either in directional cosines or Euler angles. As many as 50 data points per second can be digitized and sent to the host computer or terminal.

USAGE

The 3SPACE Digitizer is first and foremost a tool to improve productivity. It achieves this by the rapidity with which it digitizes objects and the steps it saves in the design process (see Fig. 2).

Using the 3SPACE Digitizer, designers of three-dimensional shapes (whether they be components, assemblies, or machinery) can swiftly produce realistic images of preliminary designs. In most cases, the image will be of sufficient detail to improve the design and to simulate performance.

Some 3SPACE Digitizer applications of particular interest to computer-aided modeling include:

o Building Three Dimensional Data Bases - The 3SPACE unit digitizes solids in real time for input to computer-generated imagery systems and, therefore, it eliminates the time consuming photogrammetric and model sectioning techniques that are necessary to reduce a model to two dimensions. Because the system's magnetic fields penetrate all low-conductive materials, it is not limited by interference, blockage, or noise difficulties. While other X, Y, Z digitizers require a clear line of sight between sensors and detectors, the 3SPACE Digitizer operates equally well among various spatial layouts and component shapes. Conversely, the system can function as a guide by directing its stylus to specified locations on the model.

o Generating Engineering Drawings - The 3SPACE Digitizer enables users to produce isometric drawings directly from three-dimensional models. By automating the measurement of piping and component dimensions, the unit produces significant labor savings over manual input techniques.

- o Animating Graphics - The 3SPACE Digitizer allows users to interact with computer-generated imagery. Its six-degree-of-freedom stylus can be used to direct the real-time movement and rotation of resident images.

- o Creating Video Art - The 3SPACE Digitizer can create art as a function of the orientation and position of its stylus. The stylus can also be used as a paint brush, knife, or chisel to create video painting or sculpture.

These applications are by no means inclusive. They are presented here to demonstrate the unit's versatility. In general, the system has application wherever designers contemplate doing in three dimensions what is presently restricted to two dimensions.

CONCLUSION

The 3SPACE Digitizer presents new opportunities for interacting with computers in a large number of engineering and scientific applications. Developed from proven military technology, the 3SPACE Digitizer significantly simplifies three-dimensional digitization and overcomes many of the problems that once plagued such procedures. Its ability to digitize solid objects makes it a valuable new tool for computer-aided design and manufacturing.

BIBLIOGRAPHY

1. Board, G.; and Scully, J. T.: "Communicating With Computers in Human Terms." Proceedings of Graphics Interface, 1982, National Research Council of Canada, Toronto, Canada, May 1982, pp. 329-335.

2. Bolt, R. A., "Put-That-There: Voice and Gesture at the Graphics Interface." Comput. Graphics, Vol. 14, No. 3, July 1980, pp. 262-270.

3. Raab, F. H.; Blood, E. B.; Steiner, T. O; and Jones, H. R.: "Magnetic Position and Orientation Tracking System." IEEE Transactions on Aerospace and Electronic Systems, Vol. 15, No. 5, September 1979, pp. 709-718.

4. Schmandt, C.: "Interactive Three-Dimensional Computer Space." Paper presented at the SPIE Conference on Processing and Display of 3-D Data, San Diego, California, August 22-27, 1982.

5. Schmandt, C.: "Some Applications of Three-Dimensional Input." M. S. Thesis, Massachusetts Institute of Technology, 1980.

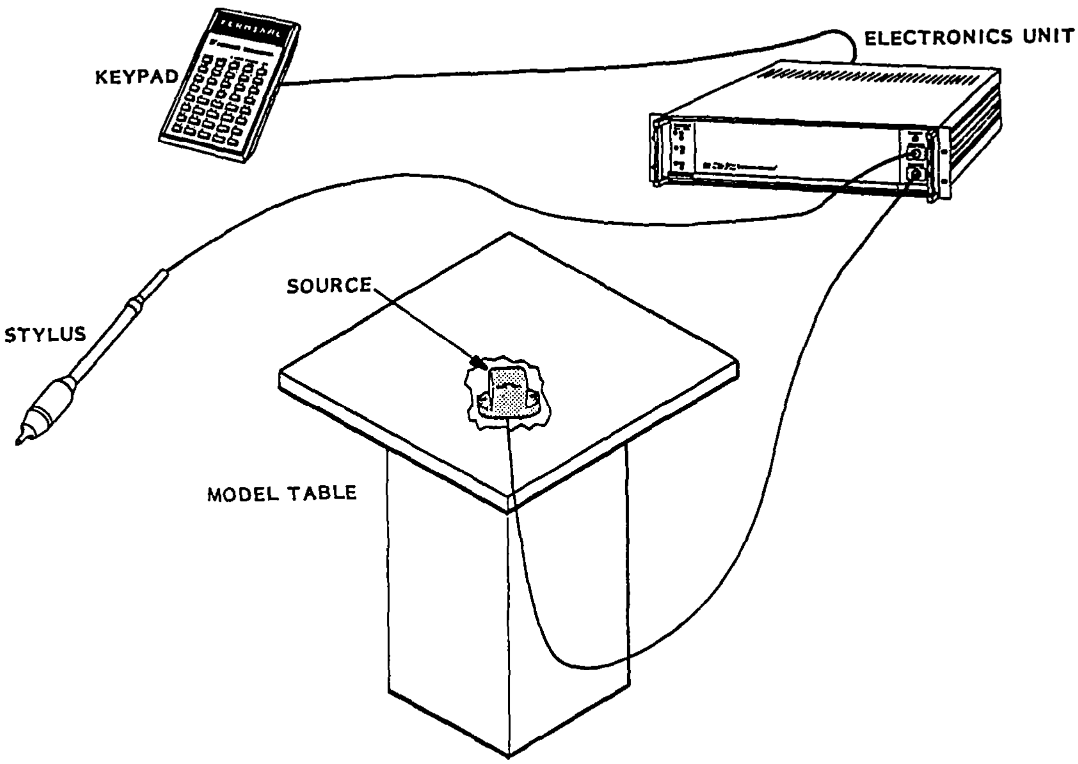


Figure 1.- 3SPACE Digitizer components.

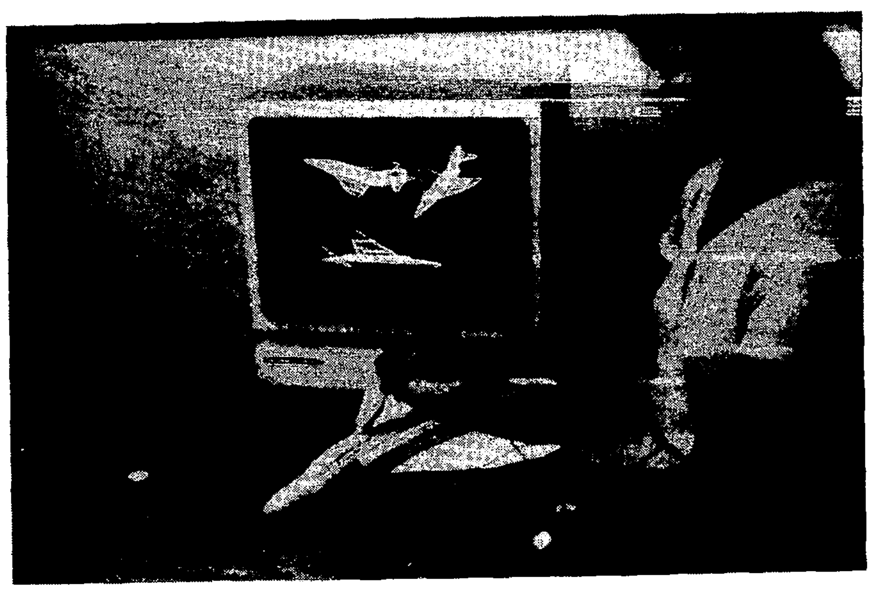


Figure 2.- Digitization of three-dimensional plastic model.

INTERACTIVE MODELING USING THE PATRAN-G PROGRAM

Winifred A. Stalnaker
NASA Langley Research Center
Hampton, Virginia

PATRAN-G is an interactive finite element generating and graphics display program written by PDA Engineering, Santa Ana, California. The program is currently executable on the VAX or PRIME computers and is compatible with 19 commercially available graphics terminals. The PATRAN program can be used in two modes: (1) generation of a finite element model, or (2) postprocessing of output data from a finite element analysis program.

The finite element generating mode is divided into two phases. Phase I is the geometry construction. During this phase, the geometric boundaries of the model are defined and the material properties are input. Mirror imaging, translation and scaling of lines, surfaces, and volumes, and rotation of regions permit the geometry to be described efficiently. Points may be defined in alternate rectangular, cylindrical, or spherical coordinate systems. Also, PATRAN-G can calculate the intersection of lines or surfaces. This capability allows a user to easily model, for example, the intersection of a cylinder which cuts though a sphere at an angle.

Once the geometry of the model has been described in terms of points, lines, surfaces, and volumes, Phase II is used to generate the computer finite element model. The first step is to divide each geometric entity with node points. The node points are then connected with element types specified by the user. Two-node elements may be specified as rods or beams, and optional offsets and orientation of the elements can be input. Plate elements may be specified as bending, membrane, etc., depending upon the analysis code that will be used. Degenerate elements are automatically generated in transition regions. For example, a triangle would be the degenerate element for a quadrilateral, and a wedge would be generated for a hex element.

The elements are plotted as they are generated and can be examined for a suitable aspect ratio and mesh density. If it is necessary to refine or coarsen the mesh, the existing nodes and elements can be easily deleted without affecting the geometry definition of Phase I. It is also possible to go back to the geometry definition of Phase I and reparametrize a line or region. Nonuniform parametrization during geometry definition will cause a nonuniform mesh generation during Phase II. This is useful when the model has a tapered region (for example, the planform of a tapered wing) and a uniform element aspect ratio is desired.

Once a satisfactory mesh has been generated, section properties, loads, and boundary conditions are applied. PATRAN is able to calculate a unique property or load for each element from a mathematical definition over a region. This feature is used, for example, in order to calculate the thickness at the centroid of each plate element in the model of a wing in which the thickness will vary from leading edge to trailing edge and from root to tip. Optimization, node compaction, and data base translation provide a computer finite element model ready to be converted to an analysis code. Optimization is an optional feature that renumbers the nodes according to a rms wavefront or bandwidth optimization criteria. Node compaction will remove all unused node numbers and renumber the nodes sequentially. This option is necessary for some analysis codes such as EAL to operate efficiently. PATRAN allows the user to convert the data base to a neutral ASCII file. This file can be converted by user-written translators to input data for analysis programs such as NASTRAN or EAL.

PATRAN has many graphics display capabilities that enhance the generation of the finite element model. Windowing, changing viewing angles, zooming, perspective views, and split screens are useful during geometry definition as well as mesh generation. Hidden-line plots are available for both the Phase I and Phase II models. During mesh generation, elements can be shrunk about the centroid. This allows each element boundary to be plotted and missing elements to be corrected. During the input of constraints and loads, each associated node is circled. Errors during input can be easily noted and corrected. Elements may be color coded by section property numbers or material numbers. This allows the user to identify any incorrect or missing numbers.

Other graphics features are useful during postprocessing. Analysis output data such as deformations and stresses can be input to PATRAN. This data, as well as data used during an analysis (such as temperatures and pressures), can be color coded. The model can then be plotted as color-coded finite element lines or as color-filled plates or solids. Contour lines or solid contour region plots are also available. Deformed plots can be superimposed on the undeformed structure. Contour lines can also be superimposed on the deformed structure. Carpet plots (each color code plotted at a different elevation) can be used to display data.

PATRAN has been operational at Langley Research Center for 2 years. During this time, the program has been used to create NASTRAN and EAL structural finite element models, to graphically display finite element models not generated by PATRAN, and to color code results.

PATRAN has greatly reduced the time required to generate accurate finite element models. It has been especially useful for NASTRAN models because NASTRAN does not have mesh generation capabilities. The curved surface generation and automatic calculation of section properties have also made the creation of EAL models more efficient. The neutral file system of PATRAN has made it easy to interface data to and from PATRAN.

The main disadvantage of PATRAN has been that control of the node and element numbering sequence is lost. If the node numbers are optimized and compressed, the numbers are no longer in the original order on the model. Also, some analysis codes such as EAL ignore the PATRAN element numbers and generate their own numbers in the order that the elements are input. In this case, it is confusing to relate element stresses to a plot generated by PATRAN. To avoid this confusion, the user must generate plots from the analysis program or use the postprocessing capabilities of PATRAN to display the results.

In summary, the PATRAN-G program has greatly enhanced the capabilities of NASA Langley to handle fast, efficient, interactive structural modeling. The program has been shown to be a powerful engineering tool for generating finite element models of complex structures, and its use has increased significantly over the past year.

INTERACTIVE GRAPHICS FOR QUICK-GEOMETRY MODELING

James C. Townsend
NASA Langley Research Center
Hampton, Virginia

The QUICK-geometry system (ref. 1) is a method for defining configuration shapes in completely analytical form. It was developed by A. F. Vachris, Jr. of the Grumman Aerospace Corporation for use when the analytical definition of aircraft geometry is advantageous or necessary for the solution of the flow around it. For example, in Grumman's shock-fitting supersonic finite-difference marching method known as the STEIN code (ref. 2), a complete cross-sectional surface definition must be available at any axial station where the marching step happens to fall. The STEIN code uses the QUICK-geometry system for surface definition in order to have this information readily available. It also uses the analytic first and second derivatives of the surface geometry provided by QUICK so that these do not have to be approximated by finite differences.

While the QUICK-geometry system provides a convenient and flexible method for generating configurations with completely analytical definitions, experience has shown it can be difficult to match a previously defined configuration with a QUICK-geometry definition. Therefore, the National Aeronautics and Space Administration (NASA) and other users have developed computer programs that aid in the generation of QUICK inputs (refs. 3-5). This paper describes a NASA-developed set of such programs which recently has been upgraded extensively to improve its usability and portability.

Before giving examples to show the use of these programs, it is necessary to outline how QUICK geometry works. Figure 1 shows the basic concept. The configuration being considered is enveloped by a set of "body lines" running in a general nose-to-tail direction. Most of these are on the body surface, but others lie inside or outside of it. Each body line has a literal name and is defined analytically in space by the user in terms of its horizontal and vertical projections. At any cross section, these lines define a set of points as shown in figure 1. These become the end points and slope control points of a cross-sectional model logical definition, supplied by the user, which defines the arcs making up the cross section.

A set of QUICK subroutines is incorporated into programs such as the STEIN code to calculate the surface points as required. The difficulty in manually generating QUICK inputs lies in properly defining the body lines. Because the basic system works well, it was decided to use interactive computer graphics to aid in generating the QUICK inputs rather than to replace a working system. The programs devised are the <u>QUICK</u> <u>I</u>nteractive <u>I</u>nput (QUIKII) codes and the <u>QU</u>ICK <u>I</u>nter<u>A</u>ctive <u>G</u>raphics <u>A</u>nalysis (QUIAGA) code. Figure 2 shows the relationship of these codes to QUICK. It also shows program QWRIT9, which takes digital data for cross sections of the configuration to be matched and writes a direct-access file to make these data accessible by QUIKII and QUIAGA.

The QUICK Interactive Input (QUIKII) program consists of the set of five codes accessing three direct-access files as shown in figure 3. Starting with digital data for cross sections of the configuration to be modeled analytically, QWRIT9 writes a direct-access file of this basic data. Then the <u>Q</u>UICK <u>I</u>nteractive <u>I</u>nput

Cross Section Generator (QIICSG) is used to fit straight lines and elliptic arcs to these cross sections interactively as shown in figures 4 and 5. Figure 4 shows the completion of a new cross-sectional model logical definition. The original data are shown as small x's, and the arcs fitted through them are solid lines with numbers to indicate the order in which they were done. The control points defining these arcs are shown as large +'s and are listed down the right side. Note that there was some difficulty in defining arc 4 on the wing lower surface. One definition that was tried is seen to have degenerated from an ellipse to a discontinuous hyperbola. Without the use of interactive graphics, such an error as this could have been left in the geometry model leading to inevitable disaster for a flow field program. The user of QIICSG can see the error immediately and correct it as shown. Figure 5 shows the same cross-sectional logical model being used by QIICSG to define control point locations at another x station. Here, as in figure 4, the original data, the previous locations of control points, and guidelines help in fitting the arcs. The control point locations, cross-sectional model logical definitions, and other information are saved on a direct-access file for later use.

When enough cross sections have been done to properly define the configuration, the QUICK Interactive Input CONversion (QIICON) code is run to write a file of QUICK cross-sectional inputs like that in figure 6. As shown in figure 3, this code also converts the cross-sectional direct-access data in body-line direct-access form. In effect, the control points from the cross-sectional definitions are strung together to form the body lines. These body lines are then mathematically modeled interactively using the QUICK Interactive Input Body Line Generator (QIIBLG). Figure 7 shows the result of this process for one body line. The cross-sectional control points are small triangles strung along the dashed line. Six line segments have been fitted through these points in a manner similar to the cross-sectional modeling. The other solid lines are previously done body lines drawn in for comparisons.

Figure 8 shows the modeling of another body line with the variety of segment shapes possible in QUICK illustrated. The use of interactive graphics in QIIBLG allows the user to see the shape of each body-line segment so that a proper choice can be made.

The last program of QUIKII is the QUICK Interactive Input OUTput (QIIOUT) code which reads the body-line modeling information from the body-line direct-access file and completes the QUICK input file by writing the body-line data after the cross-sectional data (fig. 6). At this point, the user has finished with QUIKII and is ready to run the Grumman QUICK code. This code prepares the data for use by the flow field program (e.g., STEIN).

The QUICK InterActive Graphics Analysis code (ref. 6) can be used to check the QUICK output file to detect or analyze any errors that may have occurred in modeling. It allows generation of overall views (fig. 9), comparisons of the final result with the original data (fig. 10), and display of control point locations at any cross-sectional location (fig. 10). QUIAGA calls the same QUICK subroutines as are called by flow field codes, so the geometry displayed is exactly the same as that to be used in computations. By using the various display options for analysis of modeling errors, the user can determine how the QUICK input should be edited, iterating as shown in figure 3 until a satisfactory geometry model is obtained.

QWRIT9, QUIKII, AND QUIAGA have been coded in FORTRAN 77 to run on the PRIME minicomputer system at NASA Langley Research Center using PLOT10 software for graphics display on Tektronix 4014 or 4114 storage tube terminals. A version of QUIAGA in Control Data Corporation's FORTRAN IV is also available. To improve the portability of the upgraded codes, the use of system-dependent extensions to FORTRAN has been minimized, and the use of the SPAR data base, which was required for the old version of QUIKII (ref. 3), has been eliminated.

Experience with QUIKII and QUIAGA in modeling the Space Shuttle and a number of other configurations has shown that they greatly facilitate the use of QUICK-geometry.

REFERENCES

1. Vachris, Alfred F., Jr.; and Yaeger, Larry S.: QUICK-GEOMETRY - A Rapid Response Method for Mathematically Modeling Configuration Geometry. Applications of Computer Graphics in Engineering, NASA SP-390, 1975, pp. 49-73.

2. Marconi, Frank; and Yaeger, Larry: Development of a Computer Code for Calculating the Steady Super/Hypersonic Inviscid Flow Around Real Configurations. Volume II - Code Description. NASA CR-2676, 1976.

3. Adams, Mary S.: Interactive Input for the QUICK Geometry System - User's Manual. NASA TM-81933, 1980.

4. Shope, Frederick L.: Simplified Input for Certain Aerodynamic Configurations to the Grumman QUICK-GEOMETRY System (A PREKWIK User's Manual). AEDC-TR-77-62, U. S. Air Force, Aug. 1977. (Available from DTIC as AD B021 195L.)

5. Shope, Frederick L.: Simplified Input for Certain Aerodynamic Nose Configurations to the Grumman QUICK-Geometry System (A KWIKNOSE User's Manual). AEDC-TR-77-89, U.S. Air Force, Feb. 1978. (Available from DTIC as AD A051 425.)

6. Townsend, James C.: Use of Interactive Graphics to Analyze QUICK-Geometry. NASA TM-83234, 1981; and Supplement to NASA TM-83234, 1982.

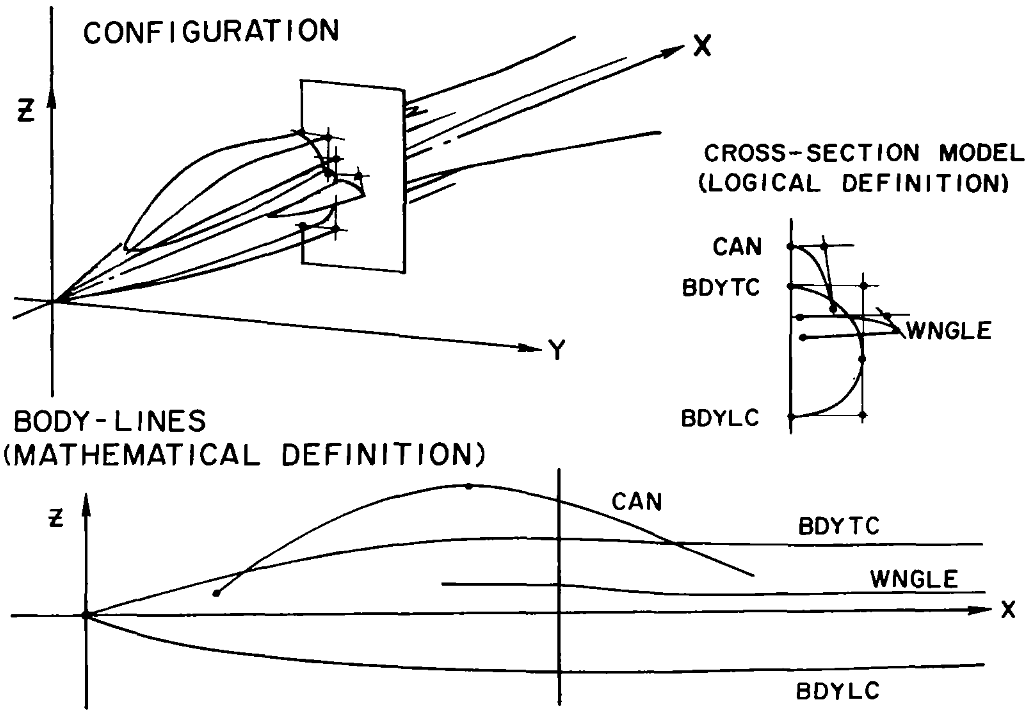


Figure 1.- Concepts used by the QUICK-geometry system to analytically model an aircraft configuration.

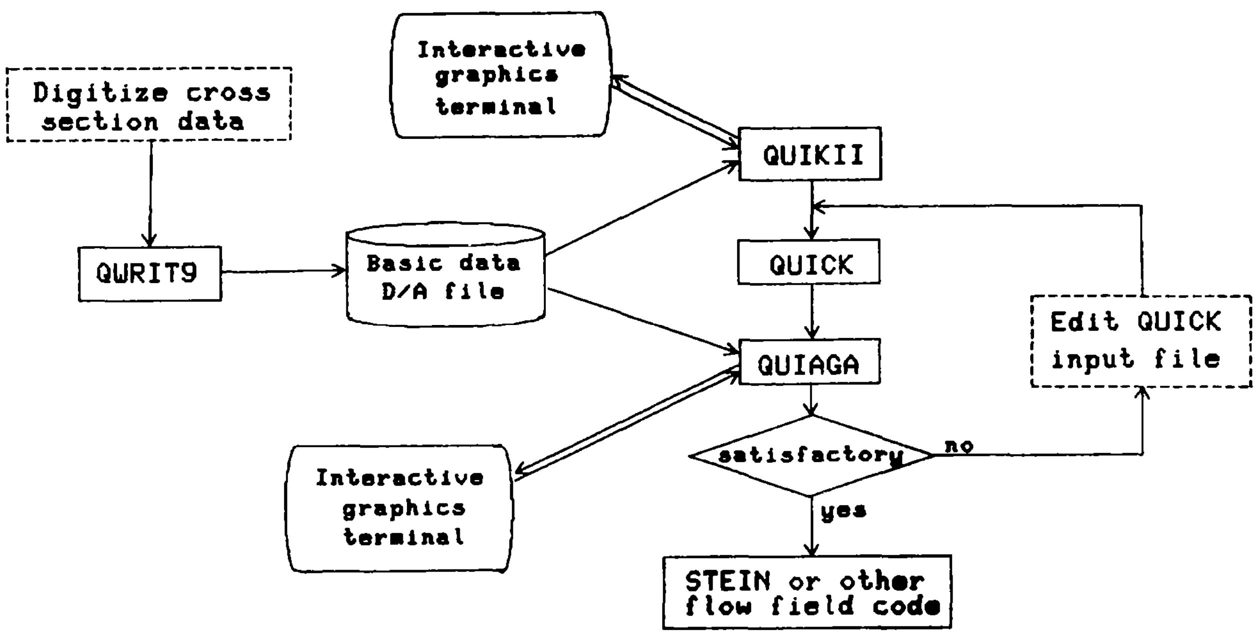


Figure 2.- Relationships between programs QWRIT9, QUIKII, QUIAGA, and Grumman's QUICK-geometry program.

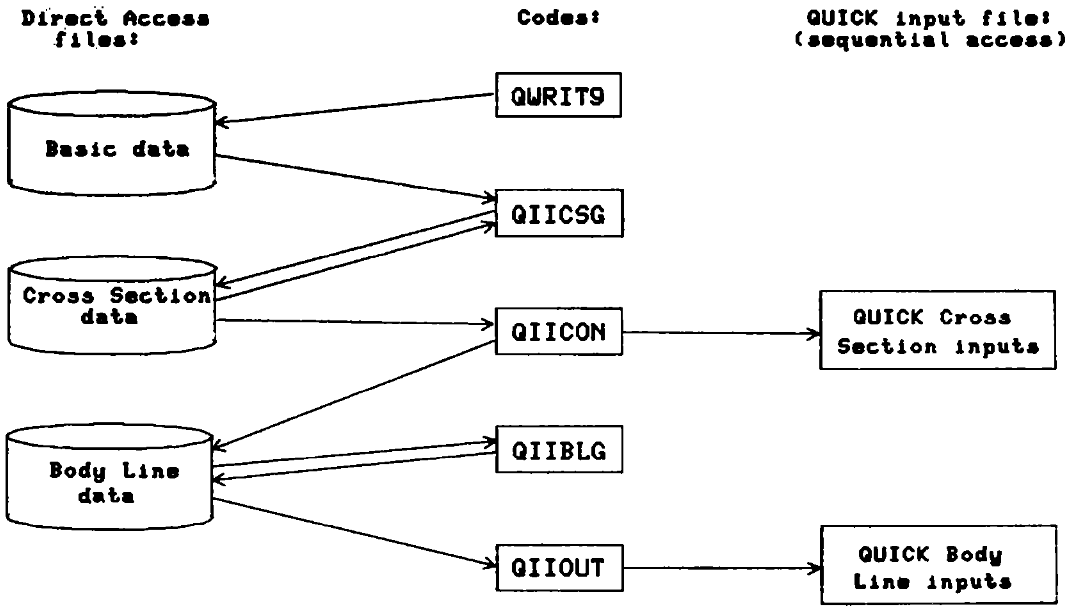


Figure 3.- QUIKII codes and the files they read and write.

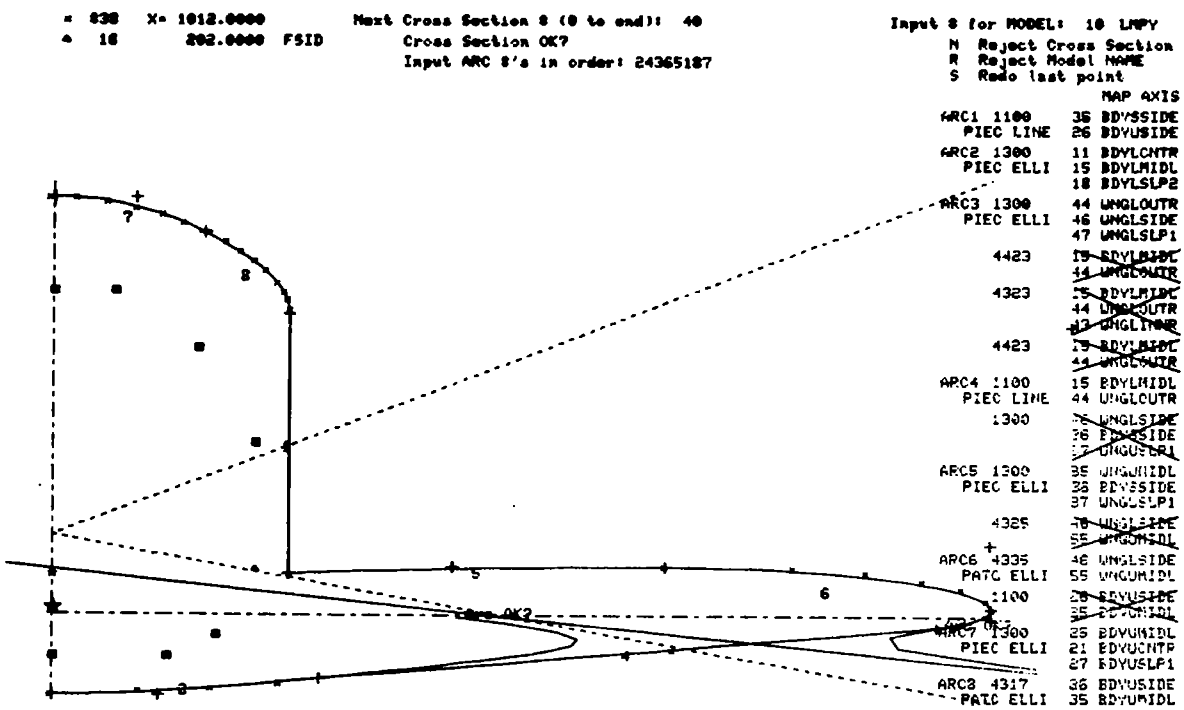


Figure 4.- Use of QIICSG to generate a new cross-sectional logical model with several attempts required to fit the wing lower surface properly.

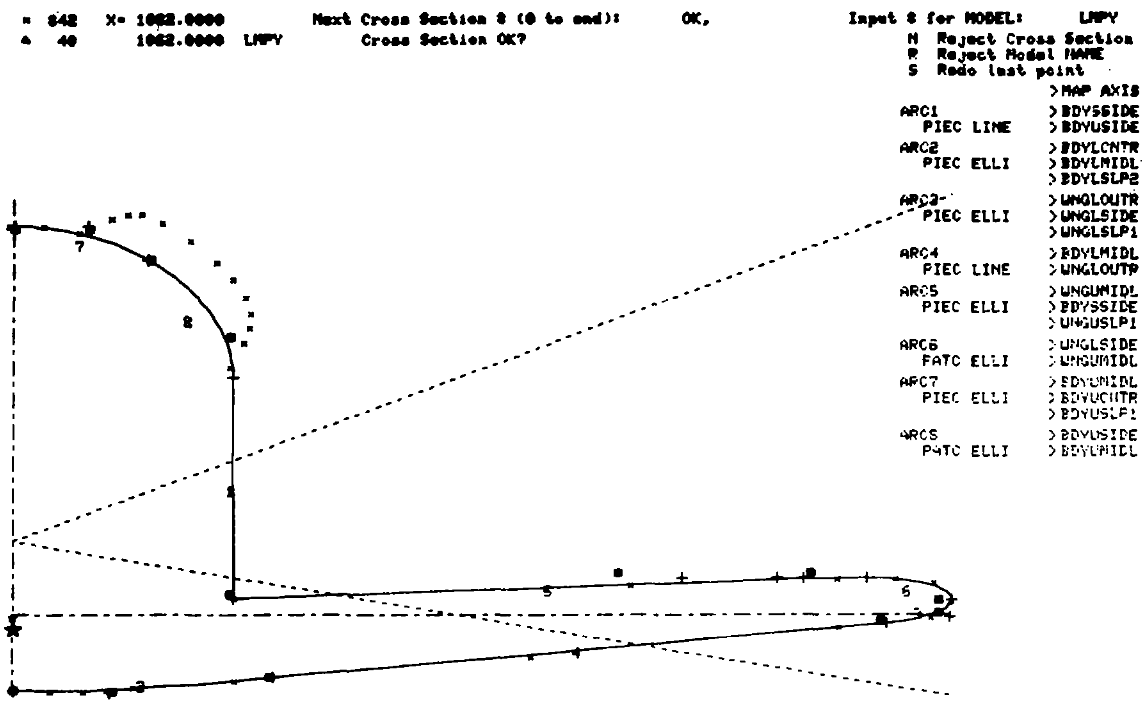


Figure 5.- Use of QIICSG to locate control points for an axial station behind that of figure 4 but with the same cross-sectional logical model.

```
CCU CONFIGURATION 01/83                                                QUICK INPUT02/083
5
1 3          3ELL
ARC1      1LINE  PIEC      BDYLCNTR  BDYLTNGT
ARC2      2ELLI  ALIN      BDYLTNGT  BDYUSIDE  BDYLSLP1
ARC3      3ELLI  ALIN      BDYUSIDE  BDYUCNTR  BDYUSLP1
2 5           SHRP
ARC1      1LINE  PIEC      BDYLCNTR  BDYLTNGT                    Cross section inputs
ARC2      2ELLI  ALIN      BDYLTNGT  BDYLMIDL  BDYLSLP1
ARC3      3ELLI  ALIN      BDYLMIDL  BDYUSIDE  BDYUSLP2        (logical model definitions)
...        ...    ...        ...       ...       ...
ARC8      8ELLI  ALIN      BDYUSIDE  BDYUMIDL  BDYUSLP1
ARC9      9ELLI  ALIN      BDYUMIDL  BDYUCNTR  BDYUSLP1
AR10     10ELLI  PIEC      CNPYINNR  CNPYCNTR  CNPYSLP1
5          MAP AXIS
1 1          0.00000   1.00000
2 2          1.00000   4.16800
3 3          4.16800  11.50000
4 4         11.50000  13.50000
5 5         13.50000  15.57000
YBDYLCNTR YUNGUSLP2
YBDYLTNGT
1 ELLX     PIEC  KV0
  0.0182    0.0000     1.0000     0.0974     0.2182     0.0056
2 CUBI     ALIN  KV0
  1.0000    1.0000     4.1680     0.6869     2.6364     0.2500      Body line inputs
3 ELLX     ALIN  KV0
  2.0000    2.0000    14.0364     1.2944     7.9455     1.2944    (mathematical definitions)
4 LINE     PIEC  KV5
 14.0364    1.2944    14.6500     1.6974     0.0000     0.0000
5 LINE     PIEC  KV5
 14.6500    1.6974    15.5700     1.6951     0.0000     0.0000
-1
YBDYUSIDE
1 ELLX     PIEC  KV0
  0.0364    0.0056     4.3273     1.1278     0.1455     0.3167
2 ELLX     ALIN  KV0
  ...        ...        ...        ...        ...        ...
```

Figure 6.- Part of a QUICK input file written by QIICON and QIIOUT from the cross-sectional and body-line data bases generated by QIICSG and QIIBLG.

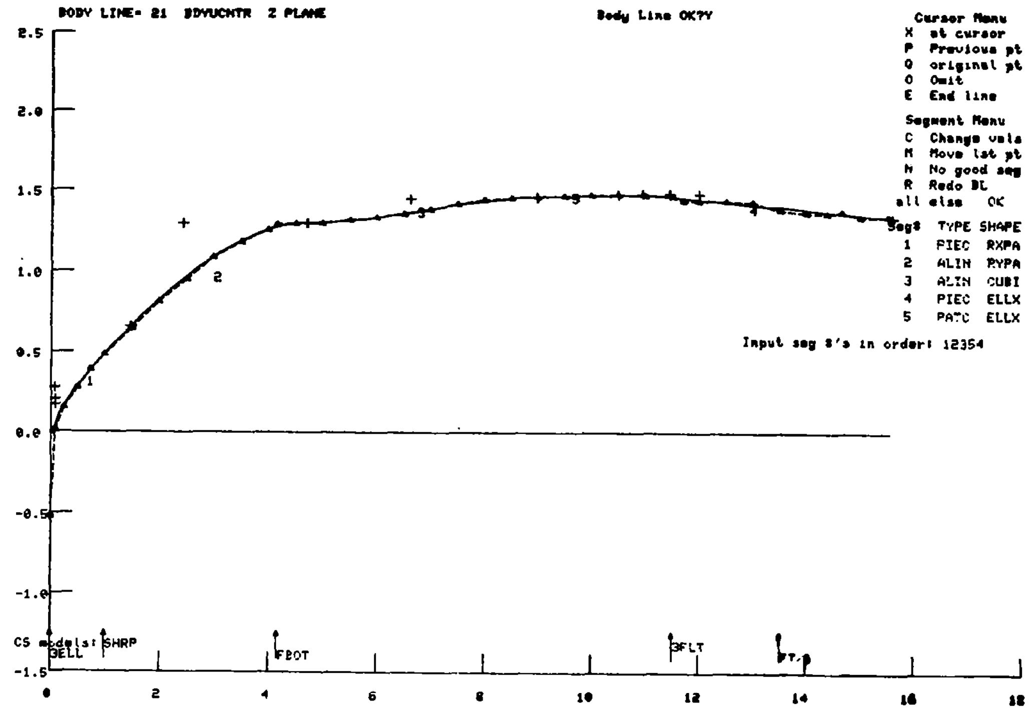


Figure 7.- Example of body line as modeled using QIIBLG.

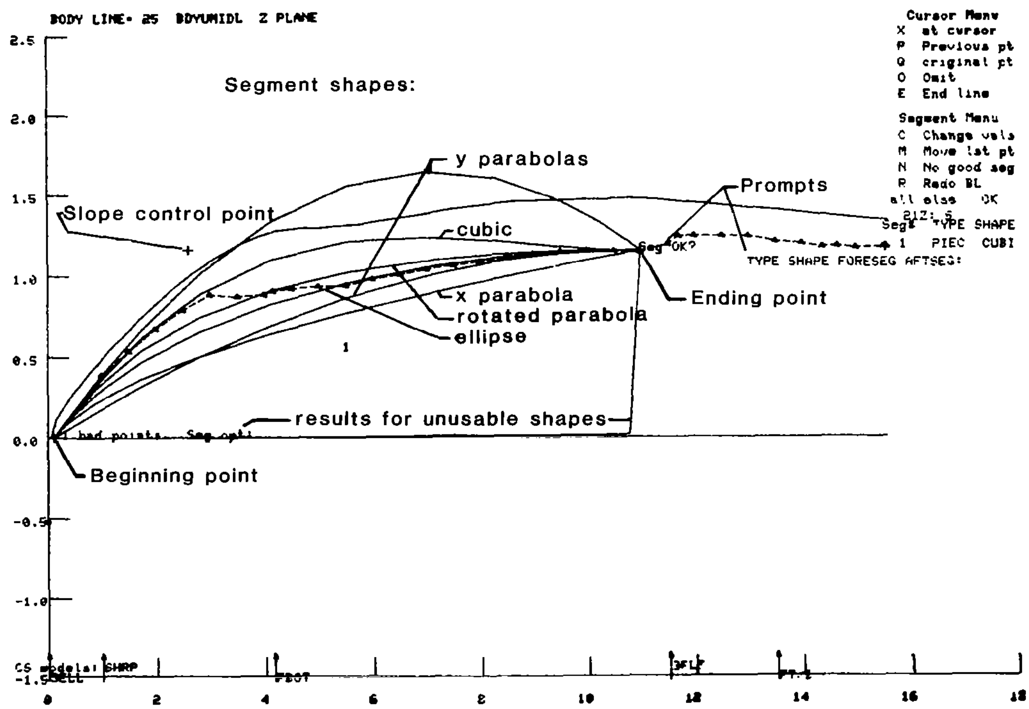


Figure 8.- Examples of body-line segments available in QUICK showing usefulness of interactive graphics in helping to make best choice. (This is not a realistic attempt at modeling.)

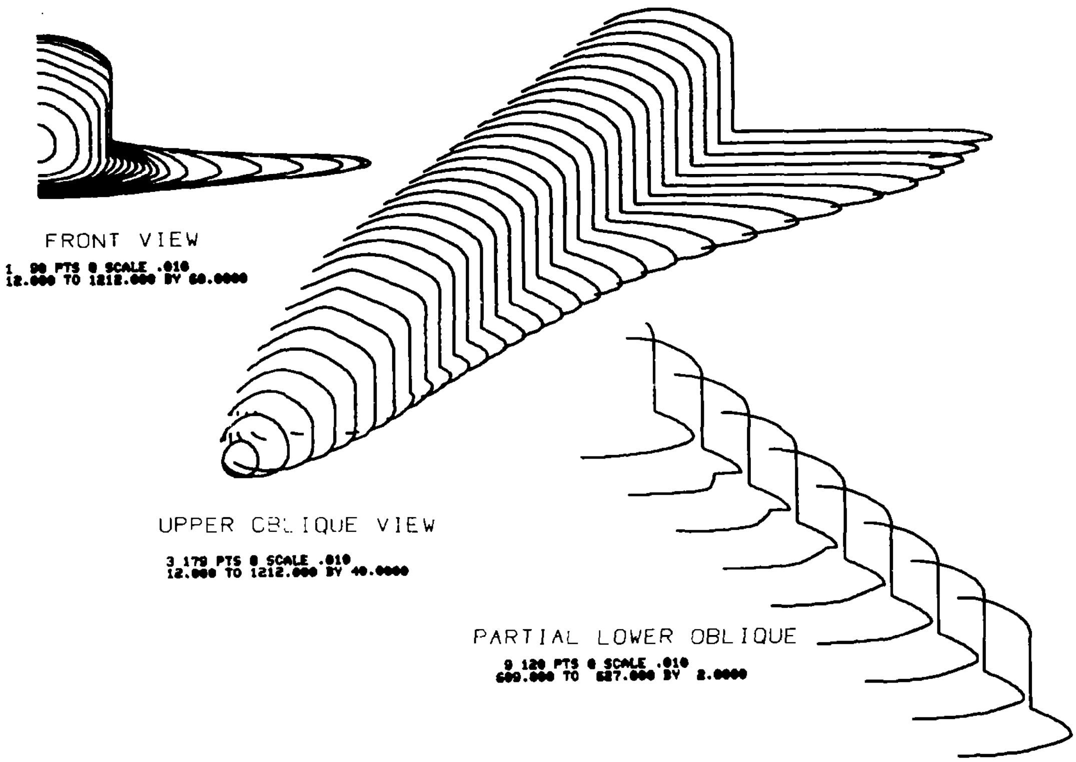
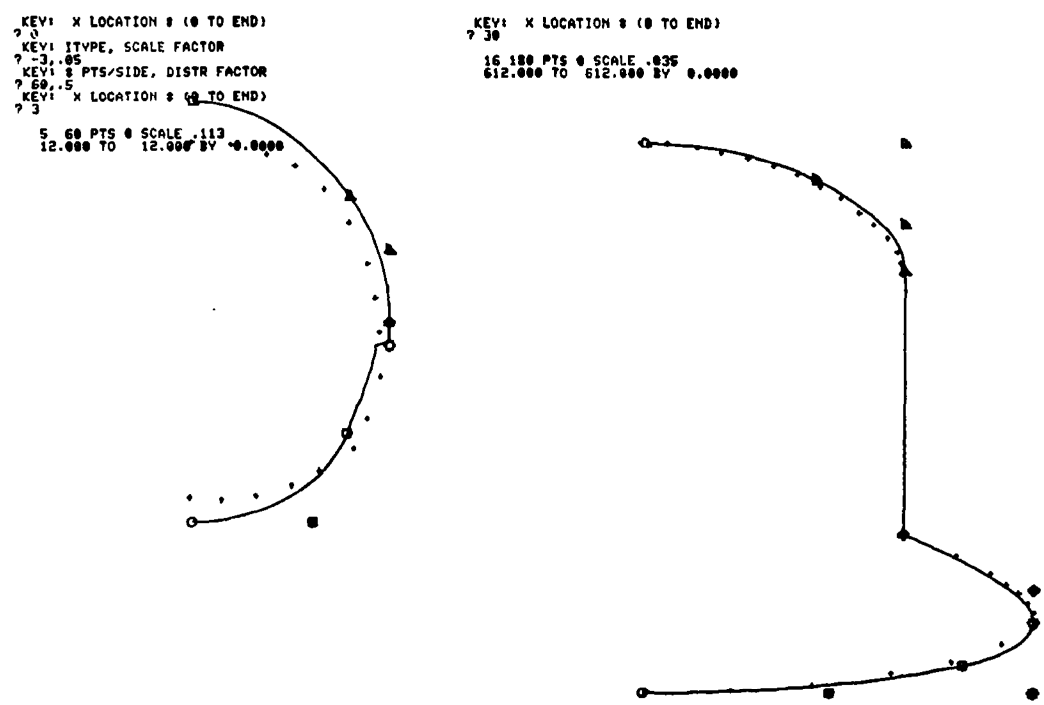


Figure 9.- Examples of views using many cross sections available in QUIAGA.



(a) Cross section in the nose region.

(b) Cross section with wing beginning.

Figure 10.- Comparisons with original data and display of control point locations using QUIAGA.

DATA ENGINEERING SYSTEMS: COMPUTERIZED MODELING AND DATA BANK CAPABILITIES FOR ENGINEERING ANALYSIS

H. Kopp, R. Trettau, and B. Zolotar
Technology Development of California
Santa Clara, California

The TDC Data Engineering System (DES) is a computer-based system that organizes technical data and provides automated mechanisms for storage, retrieval, and engineering analysis. The DES combines the benefits of a structured data base system with automated links to large-scale analysis codes.

While the DES provides the user with many of the capabilities of a computer-aided design (CAD) system, the systems are actually quite different in several respects. A typical CAD system emphasizes interactive graphics capabilities and organizes data in a manner that optimizes these graphics. It is intended for the designer who is continually making extensive revisions. On the other hand, the DES is a computer-aided engineering system intended for the engineer who must operationally understand an existing or planned design or who desires to carry out additional technical analysis based on a particular design. The DES emphasizes data retrieval in a form that not only provides the engineer access to search and display the data but also links the data automatically with the computer analysis codes.

The Nuclear Plant Data Engineering System (NPDES) is a demonstrable application of the DES concept to the nuclear power industry. For a nuclear power plant, the NPDES data bank contains, in one place, a physical plant description, component performance descriptions, operating data, and analytical results.

For convenient access, the data bank is structured in a hierarchical tree. The upper levels of the tree are organized by "engineering" systems and components so that engineers can rapidly locate and use the data in ways already familiar to them. As the user proceeds through the hierarchical index, increasing amounts of detail on the data structure are available.

Display mechanisms form an essential part of the NPDES allowing the user to rapidly understand and digest the large amount of data potentially available.

For many geometrical descriptions, integral quantities are most useful (e.g., elevations, volumes, areas, masses). However, the DES can also contain three-dimensional geometric descriptions of key components. The system has the ability to construct two-dimensional slices of such components from these descriptions. These slices may be displayed graphically (figure 1) or operated upon numerically to calculate common area intersections of two such slices.

The NPDES provides a design description of the plant for the user who prepares input to system analysis codes. First, model-making routines are used to display "cartoons" (figure 2) of the component geometries. Based on the user's engineering decisions, a nodalization is developed on the screen, and these model-making routines then use the data bank to automatically prepare the data inputs required by the analysis codes.

The automatic data preparation feature allows a user to perform complex modeling tasks in a span of minutes or hours compared to manual input preparations that previously took weeks or in some cases months. Complete model generations that previously took up to six months to develop can be accomplished within one to two days using the NPDES.

In the past, the manual storage of large volumes of geometrical piping data caused serious difficulties in using the data for inspection, modification, and reverification of power plant systems. The NPDES solves this problem by also automating the storage and retrieval of piping data. Reverification of piping data based on the computer-stored information is accomplished more rapidly and accurately than by using manually stored engineering drawings and sketches.

The NPDES includes the capability to store the actual physical piping descriptions, the piping physical location, data on the individual pipe sections, angles, valves, and pumps, along with a complete bill of materials. From this stored piping data, isometric drawings of piping runs are displayed with one of several spatial orientations as selected by the user. The drawings are also labeled based on user selections.

In addition, the stored piping data can be used to automatically prepare the input for a piping structural analysis code. This allows for complete and rapid piping analysis of both projected design changes and current configurations.

For model making, the NPDES generates piping "cartoons" (illustrated by figure 3) accessing exactly the same data used for presenting isometric displays. The engineer again nodalizes the piping system on the screen, and then model-making routines use the data bank to automatically prepare the inputs required by the analysis codes.

The DES concept and software, including the geometric capabilities, are general and can be applied to a variety of application fields. They provide the potential for a significant increase in engineering productivity.

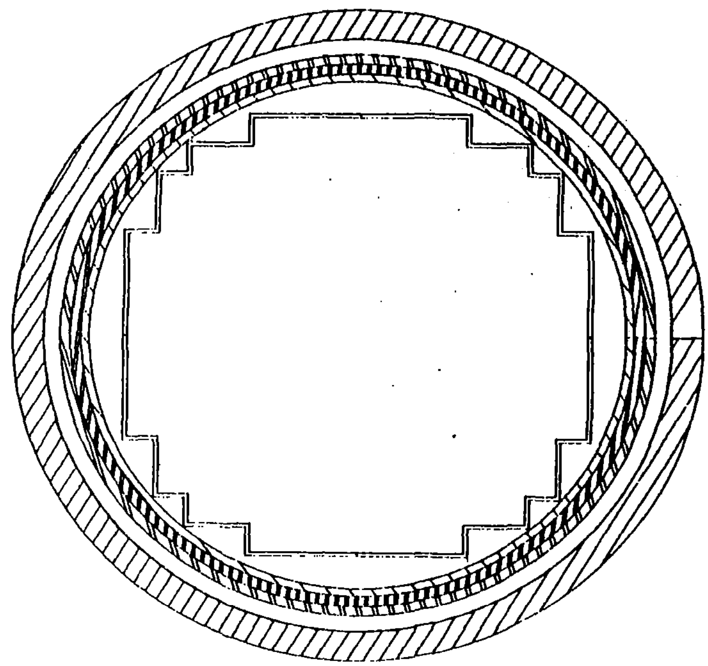


Figure 1.- NPDES graphical representation of a two-dimensional geometric slice.

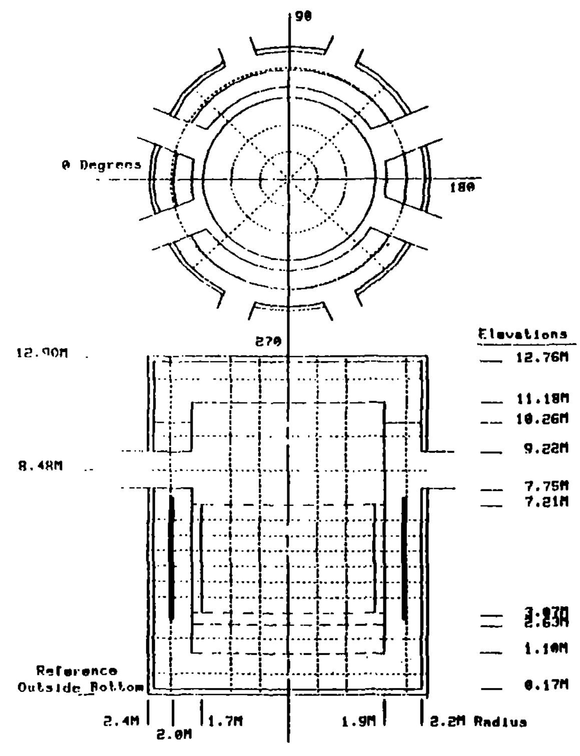


Figure 2.-NPDES component nodalization.

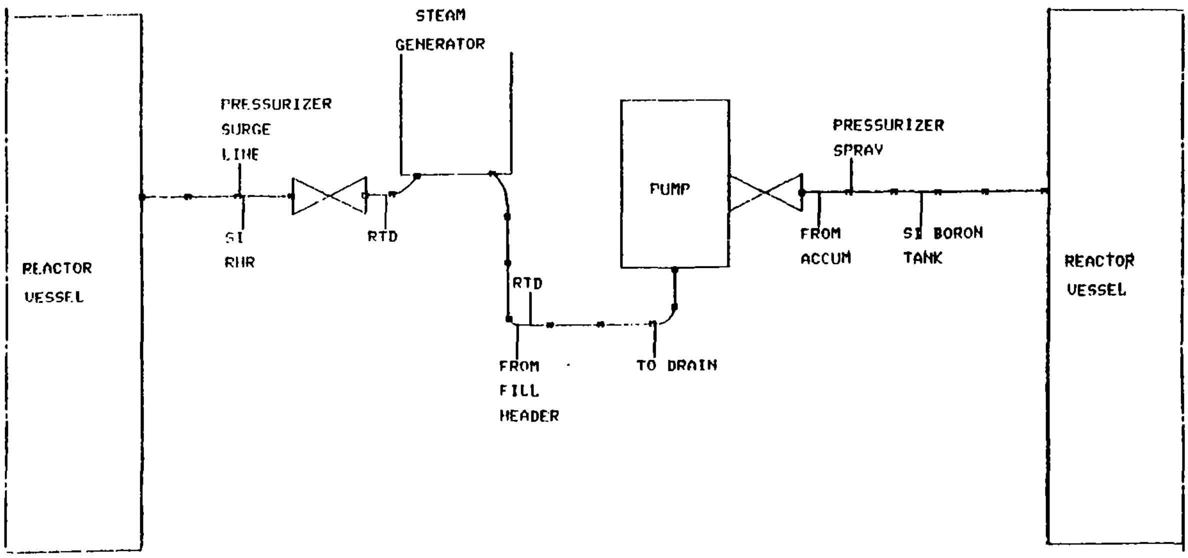


Figure 3.- NPDES piping nodalization.

SOME MATHEMATICAL TOOLS FOR A MODELLER'S WORKBENCH

Elaine Cohen

Department of Computer Science
University of Utah*
Salt Lake City, Utah

1. Motivation

It has been an important function within our research group in computer-aided geometric design to engage in some modelling exercises of particular interest to us and usually to a sponsor. From such activities come several benefits such as learning from direct experience that modelling is very often not an immediate application of theory. It is also an opportunity to learn what specialized tools and schemes might be best suited for a specific application. Taking on a specific modelling task is the true test of the effectiveness of a system or an approach to that particular class represented by the example.

This paper was written retrospectively at the conclusion of a modelling effort involving the generation of a computer model from informal drawings and a plastic model of a helicopter. When the exercise was begun at the suggestion of a sponsor at the Army Ballistic Research Laboratory in Maryland, we accepted the task as a useful target for a significant class of objects in which lofting was the predominant, characteristic modelling technique. Ships and airplanes come to mind when one mentions lofting as a technique because these objects have traditionally defined surfaces (that is, hulls and fuselages respectively) from vertical station cuts perpendicular to the vertical center plane defining the major axis of reflective symmetry. Figures 1 and 2 show a turbine blade from a jet engine that has been modelled in this way. While the aerodynamic portion fits this paradigm, the root comes from a different paradigm. Figure 2 shows the union of these two parts into a coherent model. The helicopter falls into this class, although it has characteristics of its own.

This particular exercise was one of defining the shape of the helicopter from several cross-sectional shapes, a toy plastic model of a similar helicopter, and some other knowledge about helicopters. The rotor, tail, skids, and horizontal stabilizer were designed from generic information about those parts. The rest of the model used a general form of a lofting paradigm in conjunction with Boolean operators.

There was no tangent information present other than that the body was "smooth" (except in certain regions) and that the nose was flattened (that is, it had a tangent plane perpendicular to the center line of the model). The blending of the windshield into the body had to allow for sharp turns as well as smooth transitions.

---

* This work was supported in part by the National Science Foundation (MCS-8203692, MCS-8121750), the Army Research Office (DAAG29-81-K-0111, DAAG29-82-K-0176), and the Office of Naval Research (N00014-82-K-0351).

Information about lofting lines was not present, so decisions had to be made about surface parametrization and knot placement based on the above shape data in a globally consistent manner. The parametrizations and knot determination along each cross-sectional curve have to be constrained so that the loft lines flow smoothly from section to section. In a simple example, it is easy to see that linear lofting between two circular cross sections can lead to the expected cylinder if a usual correspondence is chosen, or it can lead to a double cone if the extreme correspondence of opposite points is chosen. Another hazard is that each cross section is represented with a distinct knot collection so that the lofting process requires the union of all knot sets. It is well known that a poor parametrization can introduce undesirable undulations, so care must be taken on this account as well.

Clearly, periodic symmetric curves were appropriate to use since the curves were intrinsically periodic and left-right symmetric. Moreover, it was desirable to be able to embed corners and planar segments since the information was smoothly changing in some locations and had corners in others.

2. Tools

In attempting a modelling process of the complexity of the helicopter or of the turbine blade shown in figures 3-8, one truly needs an interactive graphical environment that offers both a dynamic vector display and a high-quality framebuffer for displaying the corresponding raster image. The interactive fitting occurs on the vector graphics station, and surface subtleties are studied on more slowly generated raster images. Both devices communicate distinct and independently useful geometric information; both are useful in modelling complex geometries.

The first tool required to begin this modelling task is a good set of routines for reproducing the cross-sectional curves. This requires parametric periodic spline-approximation methods that allow adjustment of the parametrization and addition and alteration of (psuedo)knots. One should be able to constrain the approximation to interpolate certain data points but yet approximate others. One such capability is a parametric periodic least-squares capability so that some data reduction and noise elimination can occur before and during the curve representation process. As least squares is sensitive to knot placement and parametrization, it is very helpful to have some interactive control over these items. Additionally, the degree of continuity is a variable. Quadratics can specify the curves well, and the helicopter could then be lofted. This process looked very good on the line drawing display. However, when high-quality shading calculations were made, some curvature discontinuities were seen to be undesirable.

Once the curves are reproduced in a promising way with parametrizations that are coherent from section to section, then a lofting tool is needed. That is a routine that accepts the section curves as input and outputs a surface or "skinned" fuselage. However, in handling a model that bulges and then contracts abruptly to meet a cylindrical boom, one needs control over the degree of derivative continuity at some sections. By appropriately decreasing the continuity class at places of abrupt transition, it is possible to achieve the interpolation without introducing the undulations that would otherwise occur from standard spline interpolation.

Treating the flat nose is difficult. A piecewise-explicit parametrization is not appropriate since it would have to incorporate infinite slopes, so a parametric representation is required. A linear parametrization by the longitudinal axis parametrization is impossible for the same reason.

Finally, it was decided that the body and window features could be represented more faithfully by modelling the shape as a unified object and then cutting out a section of surface and introducing a transparent insert in its place. This requires, in the general case, an intersector that is capable of cutting the body with the extruded outline of the windows and then allows for a replacement of that surface section. Thus, the boundaries can be perfectly matched. This is an example of the general case of using Boolean operators for effecting a modelling definition.

3. Remote Viewing

In order to maintain a close visual dialogue with BRL which furnished the impetus, we have used the ArpaNet for electronically transferring generic framebuffer images. A framebuffer metafile format was developed to convey the image information in a device independent manner. Thus, an image displayed in our Graphics and CAD Laboratory on a Grinnell framebuffer can be transmitted within minutes to BRL, compiled in Ikonas code, and displayed on their monitor. Within this scheme it is still necessary to take into account that the difference in aspect ratios causes directional distortions; this is a standard problem that can be satisfactorily addressed at generation time.

4. Conclusions

During the course of this modelling effort, we were able to gain some insight into the kinds of CAGD tools that would be appropriate to support this kind of work. It is one of the expected results of a project like this that we acquire the impetus to develop a set of mathematical software tools in a "workbench environment" so that subsequent efforts to model related objects will become much more straightforward.

The Alpha_1 System, which is an experimental spline-based solid modeller used to generate the models discussed in this paper, is being developed in Computer Science at the University of Utah. It is intended to become an interactive environment that will eventually provide the designer with the kind of workbench that is necessary to carry out a large variety of tasks beyond specialized lofting projects. This workbench must contain many mathematical and algorithmic tools in addition to those mentioned in this particular discussion.

BIBLIOGRAPHY

1. de Boor, Carl: A Practical Guide to Splines. Applied Mathematical Sciences, Springer-Verlag, 1978, pp. 377-387.

2. Cohen, Elaine; Lyche, Tom; and Riesenfeld, Richard F.: Discrete B-Splines and Subdivision Techniques in Computer-Aided Geometric Design and Computer Graphics. Computer Graphics and Image Processing, vol. 14, no. 2, October 1980, pp. 87-111. Also Tech. Report No. UUCS-79-117, Department of Computer Science, University of Utah, October 1979.

3. Dietz, Paul H.: Solid Modelling at the US Army Ballistic Research Laboratory. Proceedings of the Third Annual Conference of the NCGA, National Computer Graphics Association, Inc., vol. 2, June 1982, pp. 949-960.

4. Requicha, A. A. G.; and Voelcker, H. B.: Solid Modeling: A Historical Summary and Contemporary Assessment, IEEE Computer Graphics and Applications, vol. 2, no. 2, March 1982, pp. 9-24.

5. Riesenfeld, Richard F.; Cohen, Elaine; Fish, Russell D.; Thomas, Spencer W.; Cobb, Elizabeth S.; Barsky, Brian A.; Schweitzer, Dino L.; and Lane, Jeffrey M.: Using the Oslo Algorithm as a Basis for CAD/CAM Geometric Modelling, Proceedings of the Second Annual Conference of the NCGA, National Computer Graphics Association, Inc., June 1981, pp. 345-356.

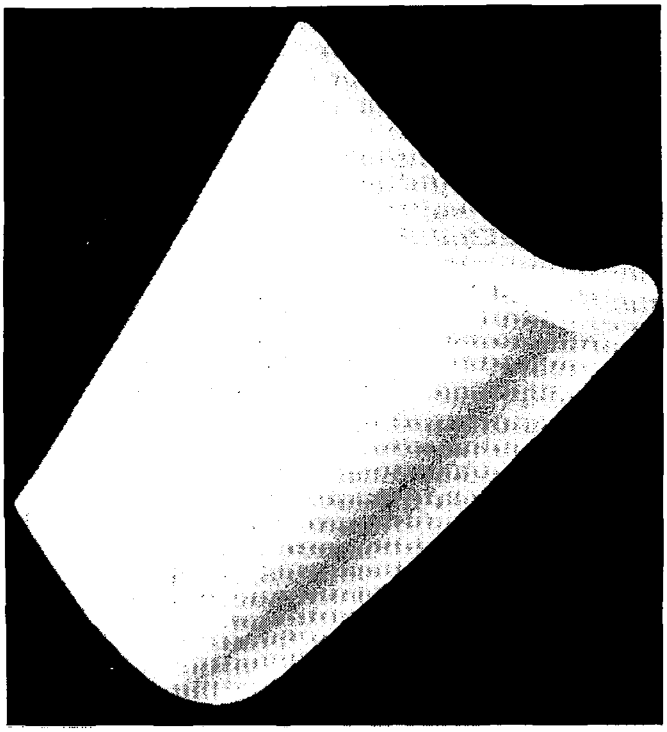


Figure 1: Lofted turbine blade.

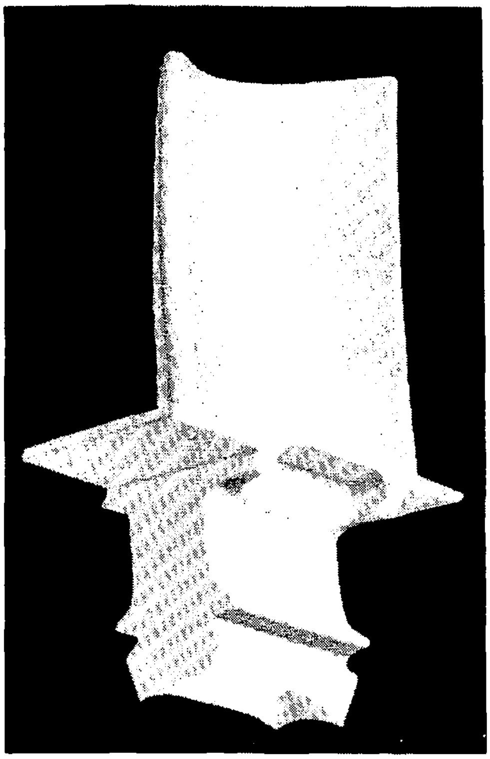
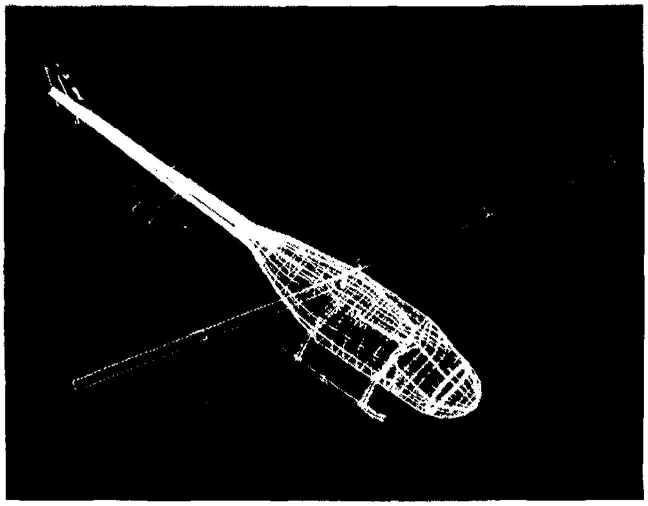
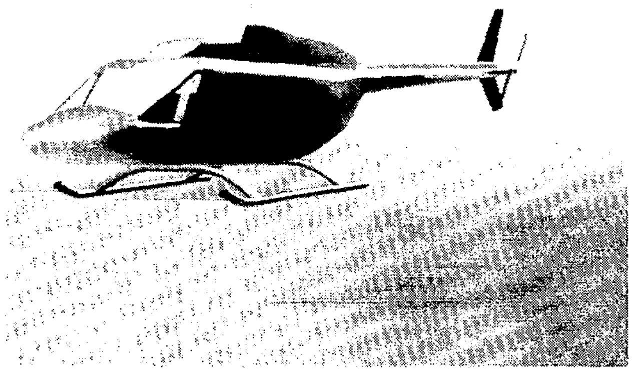
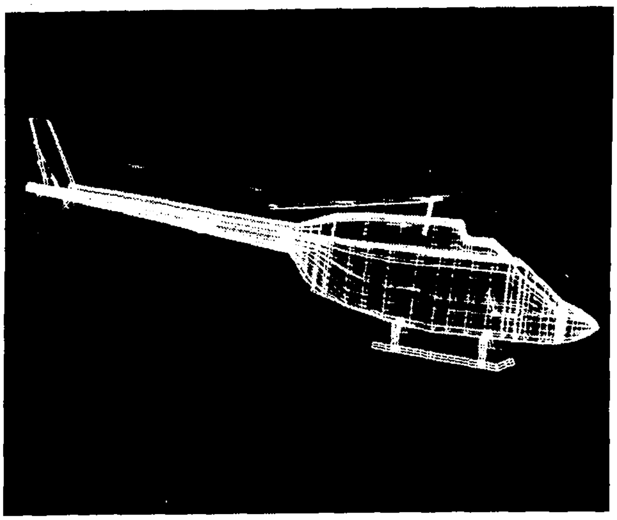


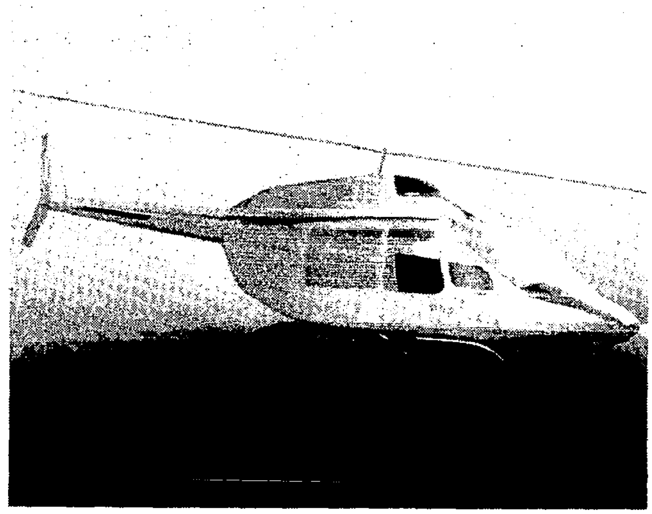
Figure 2: Completed model using Alpha_1.



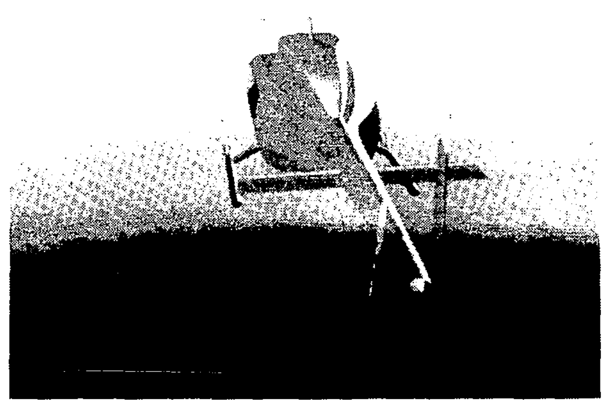


Figures 3-4: Spline-lofted helicopter modelled with Alpha_1 (top view).





Figures 5-6: Spline-lofted helicopter modelled with Alpha_1 (side view).



Figures 7-8: Spline-lofted helicopter modelled with Alpha_1 (rear view).

THE TRIANGLE COMPARE METHOD OF HIDDEN-LINE ELIMINATION

Charles R. Price
NASA Johnson Space Center
Houston, Texas

An original algorithm is presented which solves the classical computer graphics problem of eliminating those lines that should not be visible in a wire-frame model representation of a solid figure. The basis of this Triangle Compare algorithm is that any polygon, regardless of its complexity, can be constructed from a set of triangles. Thus, any solid figure that can be described or approximated by a wire-frame model (ref. 1) can also be constructed from a set of triangles the sides of which are either part of the original wire frame or are construction lines that are added when the wire-frame polygons are decomposed into triangles. In the Triangle Compare algorithm, once the triangles defining the figure have been defined, they are ordered based on the nearness of each triangle to the viewer (Z-component sort) and stored in a linked list. An overlap comparison is then made of the nearest triangle to all succeeding triangles, and those triangles or parts of triangles that are occluded from the viewer by the referenced triangle are eliminated. The remaining parts of triangles are synthesized into other triangles and are added in order to the linked list. Subsequent reference triangles are provided by a traversal of the linked list. After the entire list has been traversed and each triangle has been used as a reference, the resulting list is used for a final rendering with hidden lines removed via calligraphic MOVE and DRAW commands.

During the development of the Triangle Compare algorithm, a limited implementation of the algorithm coded for the IBM Personal Computer was used to demonstrate proof of concept on a simple wire-frame problem figure (see figures 1 and 2). A comparative evaluation was also made between the Triangle Compare algorithm and historical solutions to hidden-line elimination algorithms (ref. 2), and while the Triangle Compare method was found to be unique, some functional similarity to the Newell method was identified.

A flowchart of the Triangle Compare algorithm is seen in figure 3. The process begins with the wire-frame model. Each two-dimensional face of the figure is decomposed into triangles and stored in a doubly linked list. Each triangle consists of four world-space sets of coordinates with a MOVE or DRAW command for each set of coordinates. The command for the first set of coordinates is always a MOVE command; the remaining commands are always either DRAW SOLID or DRAW DASHED. The command SOLID is associated with the wire-frame lines, and DASHED is associated with construction lines. The first and fourth sets of coordinates are always equal.

The decomposition of the wire-frame model into triangles is straightforward in concept but nontrivial in machine implementation. The two-dimensional work of Freeman and Loutrel (ref. 3) supports an automatic mechanism, but development of such an algorithm was not included in this effort. Definition of all construction lines was determined manually for the problem figure (figure 4).

After the wire-frame model has been defined in the triangle format, any scaling, rotation, translation, or perspective transformations desired on the figure are performed and the results stored in a data structure identical to the one in which the untransformed triangle-composed figure is stored. The MOVE/DRAW commands are not affected by any of the scaling, rotation, or translation transformations. These geometric transformations produce a screen-referenced two-dimensional figure in X and Y with the Z component proportional to the apparent distance from the viewer's eye.

After the geometric transformations have been completed, each resulting triangle is internally sorted by its Z component. Identification of each nearest Z component is stored as part of the linked-list pointer structure. Next, a sort is performed among the triangles' nearest Z components. Again, the results of these sortings are reflected in the linked-list pointers. As a result of these sortings, the triangles that define the figure are ordered according to their proximity to the viewer.

Given the Z-component-ordered list of triangles defining the figure, the comparison of the triangles can now be made in screen coordinates (see figure 5). This comparison is made by successively comparing each triangle (in order starting with the triangle nearest the viewer) to all subsequent triangles in the list. The comparison is accomplished in two stages.

First, each vertex of the subject triangle is tested to determine if it falls within the boundaries of the reference triangle. To accomplish this, reference vertices and subject vertices are first tested for coresidency; then, for those subject vertices not coresident with any reference vertex, a comparison is made between the area of the reference triangle and the sum of the areas of the triangles formed by each subject triangle vertex and each side of the reference triangle. The results of this test indicate one of three possible states: (a) The subject triangle vertices are entirely outside of the reference triangle and should remain in the candidate list for final rendering. (This is valid even if the reference triangle is entirely contained within the subject triangle boundaries); (b) The subject triangle is contained within the boundaries of the reference triangle and should be eliminated from the list; (c) The subject triangle is partially within the reference triangle and should be eliminated from the list, but additional triangles should be generated from any of the parts of the subject triangle which are not contained within the boundaries of the reference triangle. These new triangles should then be added in order by Z components to the candidate list.

The second part of the comparison is the determination (for the partially overlapped triangles) of the points of intersection of the overlapping sides. This is done by solving the two linear equations that represent the intersecting sides in screen coordinates for the common intersection point and then interpolating the Z components based on the screen coordinates solution (figure 5). Identification of the intersecting lines as well as the coordinates of the intersection points are stored in reference tables.

The next and logically most complex part of the algorithm is the generation of the new triangles resulting from the overlap conditions. The complexity in the logic is due to 14 different possible modes of triangle overlap/generation. For a given comparison case, the particular mode of overlap is determined from the number of coresident vertices, the number of subject triangle vertices inside the reference triangle, the number of intersection points, and the identity of the intersecting sides. These 14 modes are described in figure 6, and isolation logic to distinguish among these modes is seen in figure 7.

After the candidate list has been traversed and all triangles currently resident in the list have served as reference triangles, the list is considered to contain only those triangles that comprise the correct solid-figure drawing with hidden lines removed. Using this resultant list, the final figure is rendered by executing all MOVE and DRAW commands and by executing MOVE for the DRAW DASHED commands associated with the triangles remaining in the list.

The effort to date has demonstrated the proof of concept of the Triangle Compare hidden-line elimination algorithm. Future plans include a performance comparison with a recently published alternative concept (ref. 4).

REFERENCES

(1) Newman, W. M.; Sproull, R. F.: Principles of Interactive Graphics. McGraw-Hill, 1973.

(2) Sutherland, I. E.; Sproull, R. F.; and Schumacker, R. A.: A Characterization of Ten Hidden Surface Algorithms. Computing Surveys, Vol. 6, No. 1, March 1974, pp. 1-55.

(3) Freeman, H.; and Loutrel, P. O.: An Algorithm for the Solution of the Two-Dimensional Hidden Line Problem. IEEE Transactions, Vol. EC-16, No. 6, December 1967, pp. 784-790.

(4) Hedgley, D. R., Jr.: A General Solution to the Hidden Line Problem, NASA Reference Publication-1085, 1982.

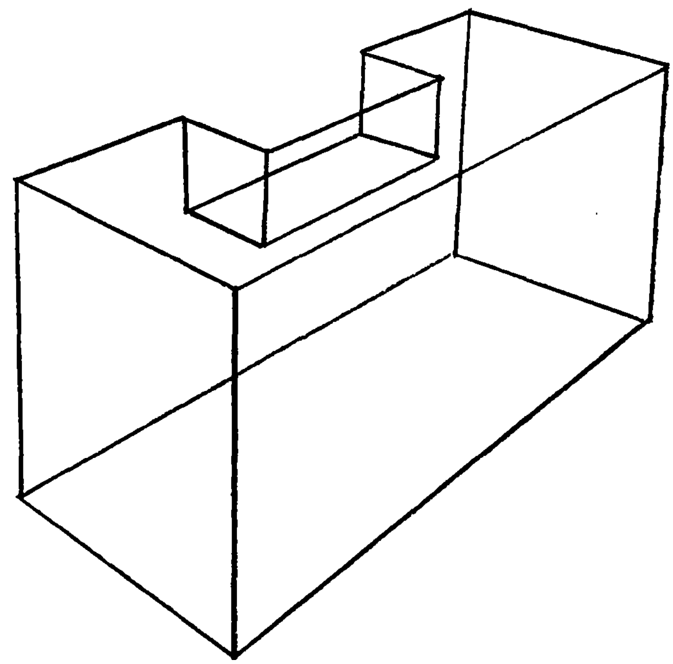


Figure 1.- Wire-frame problem figure.

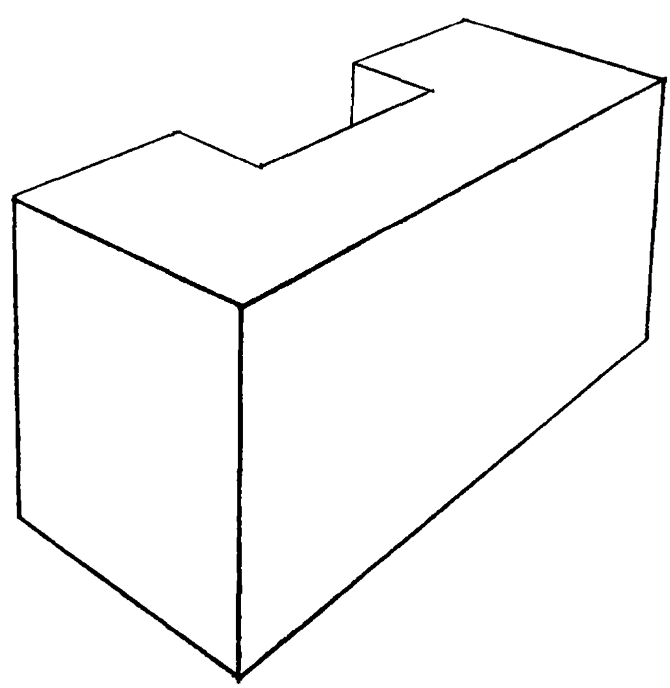


Figure 2.- Problem figure with hidden lines removed.

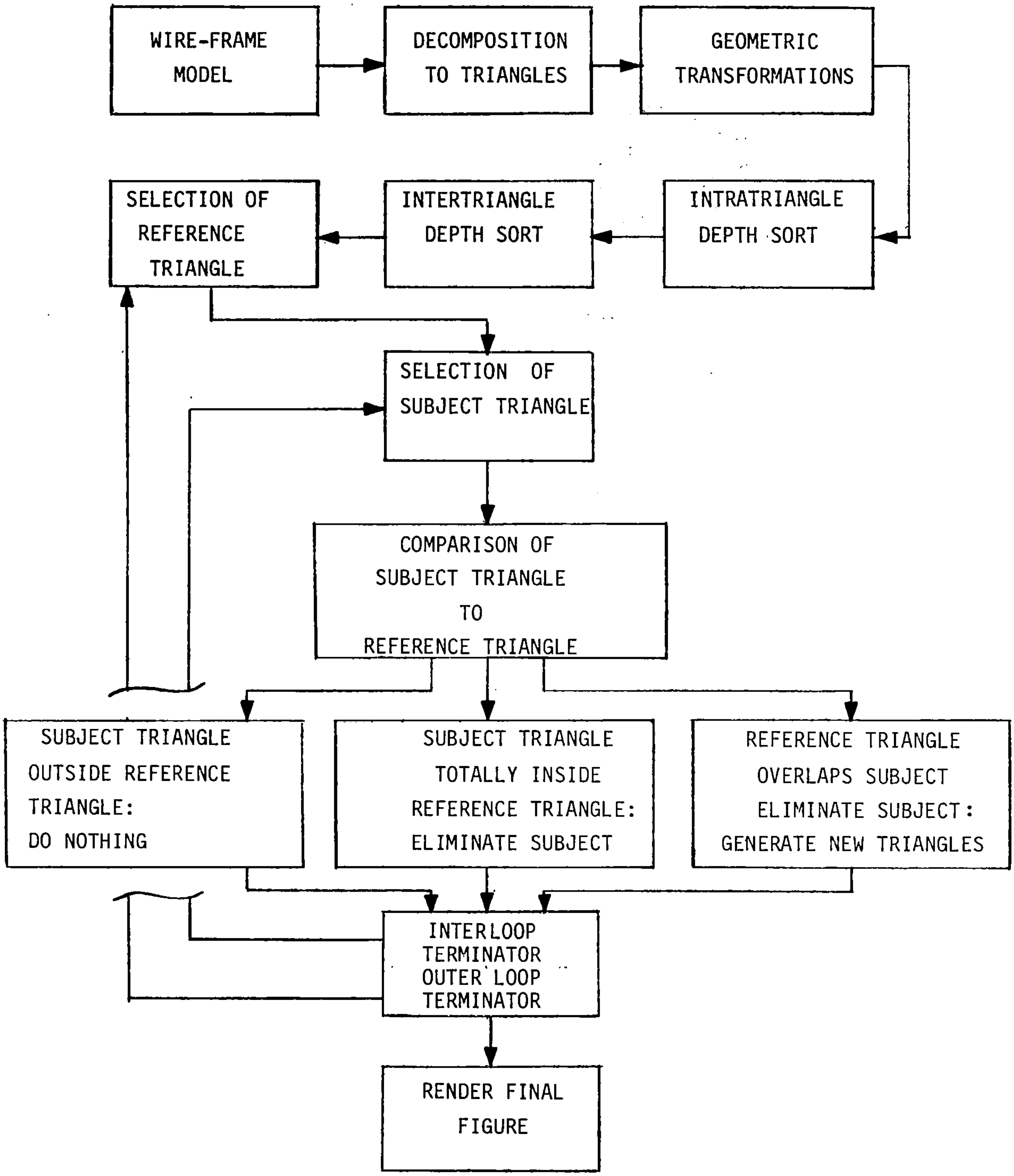


Figure 3.- Flowchart of Triangle Compare algorithm.

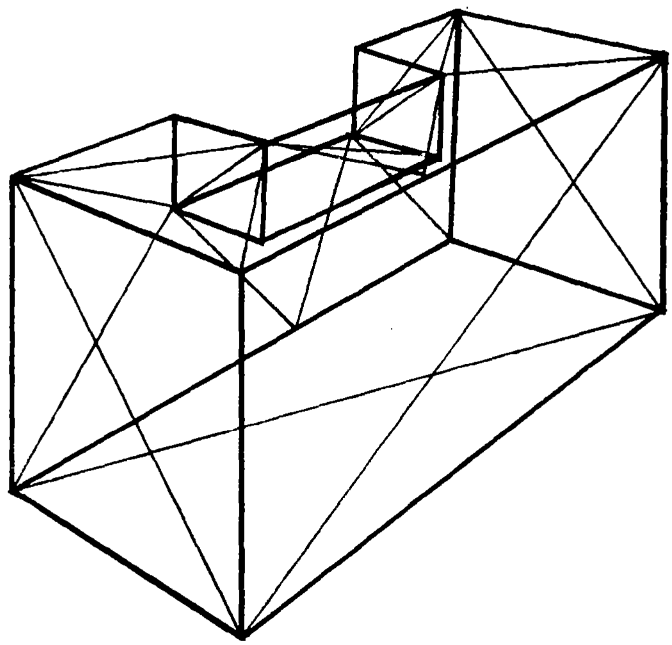


Figure 4.- Triangularized problem figure.

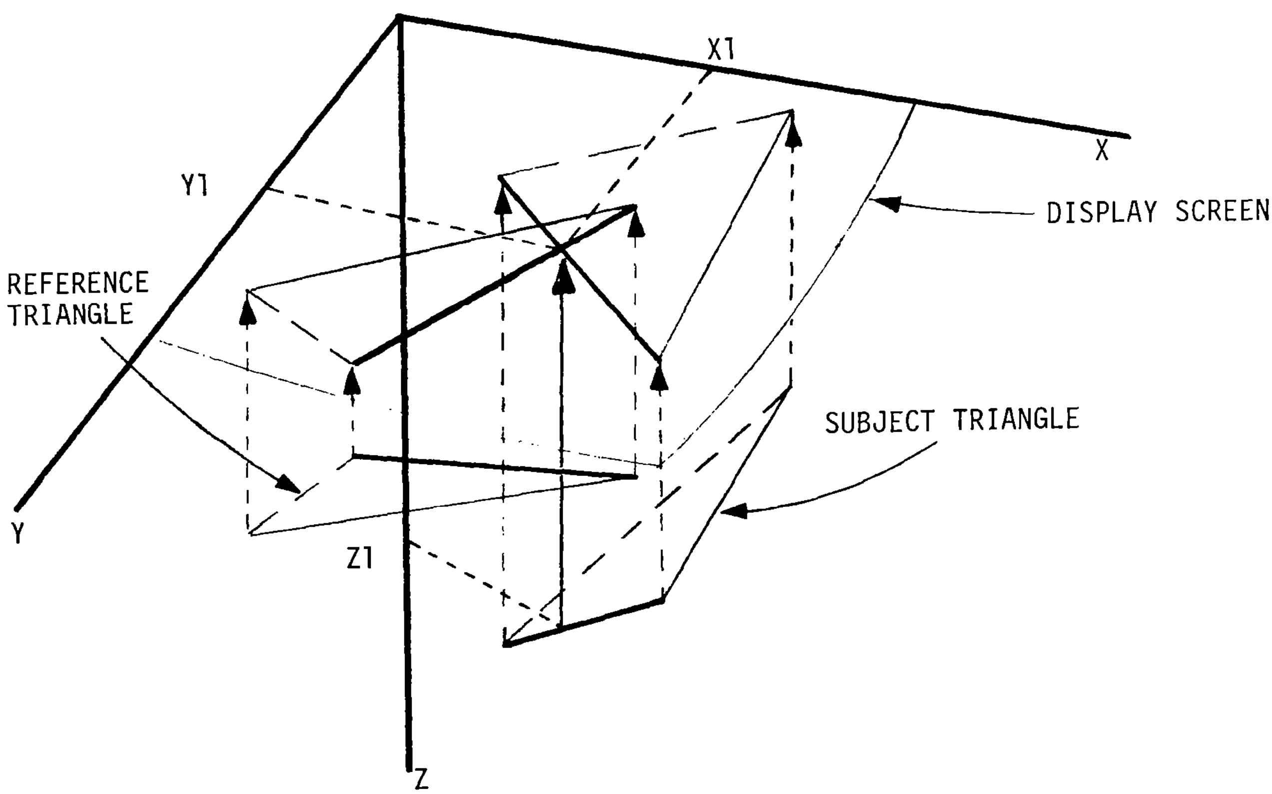


Figure 5.- Interception geometry.

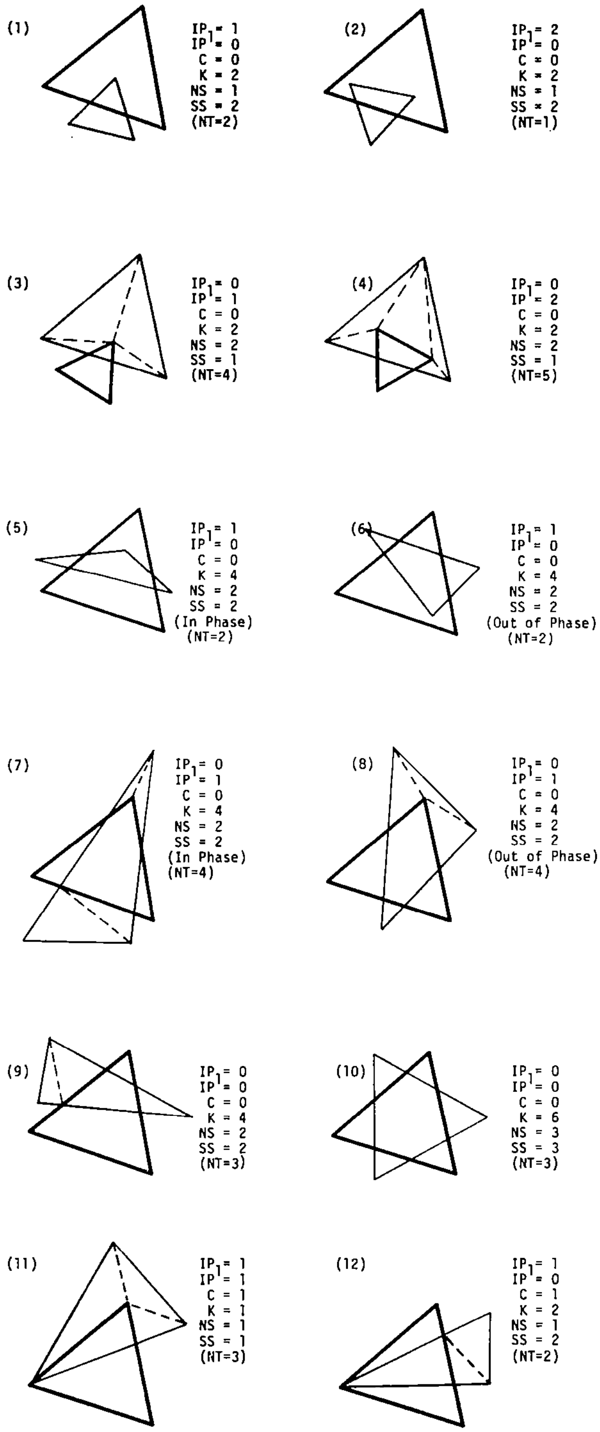
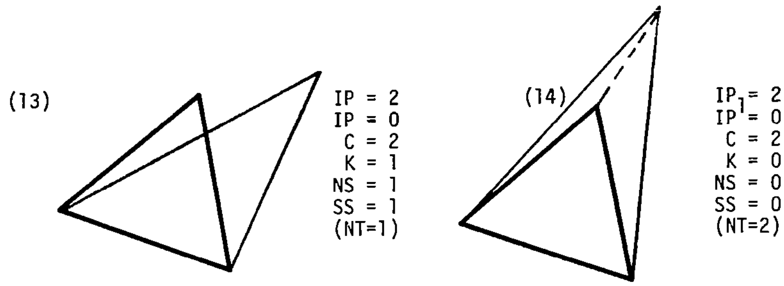


Figure 6.- Triangle overlap modes.



The 14 possible overlap modes of triangle comparison where:

IP = Number of subject triangle vertices inside the reference triangle

IP^1 = Number of reference triangle vertices inside the subject triangle

C = Number of coresident vertices

K = Number of intersections of sides

NS = Number of reference triangle sides involved in intersections

SS = Number of subject triangle sides involved in intersection

NT = Number of new triangles to be generated

Figure 6.- (Concluded).

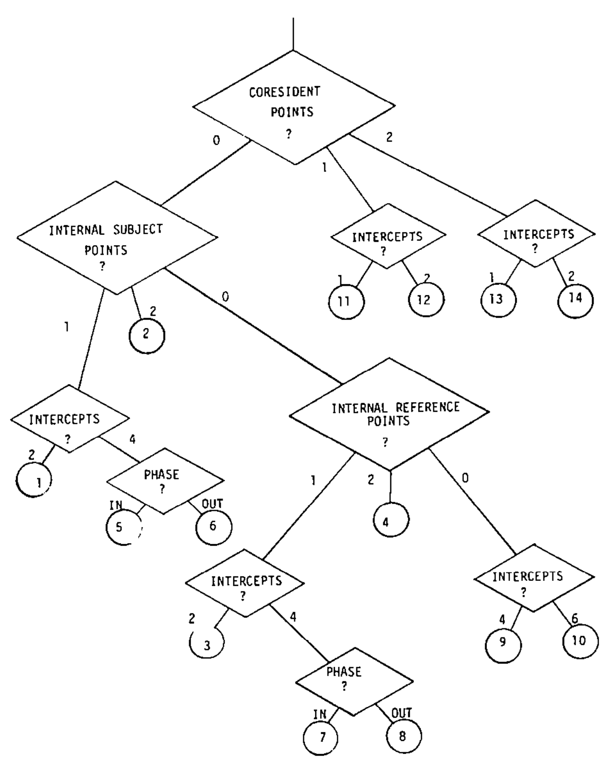


Figure 7.- Isolation logic for determining overlap mode.

COMPUTER GRAPHICS IN AERODYNAMIC ANALYSIS

J. V. Cozzolongo
NASA Ames Research Center
Moffett Field, California

Introduction

Research activities in the Aircraft Aerodynamics Branch at NASA Ames Research Center involve the prediction of aerodynamics of various aircraft concepts ranging from V/STOL fighters, shown in figure 1, to advanced turboprop configurations, shown in figure 2. These predictions are based on a combination of computed data and wind tunnel test data. The two methods for predicting the aerodynamics of aircraft produce large quantities of flow field data.

Techniques usually used in postprocessing three-dimensional data are two dimensional. Flow field data are shown in cross-sectional cuts with vectors representing velocities. Pressure data are displayed using X-Y-type plots. In both cases, these displays of the data are limiting. Only a local view of the data is presented. This can restrict understanding the overall effect of a portion of the flow on another area. A global view of three-dimensional data provides a comprehensive view of the flow field.

Computer graphics is being used as an approach to display three-dimensional aerodynamic data. A display generated using computer graphics has the ability to show a global view of three-dimensional flow field data and its relationship to aircraft geometries.

The main emphasis of this paper is to show how the use of computer graphics helps in aerodynamic analyses on a routine basis. The mathematical modelling of the aircraft geometries and the shading technique implemented will be discussed. Examples of computer graphics used to display aerodynamic flow field data and aircraft geometries will be shown. A future need in computer graphics for aerodynamic analyses will then be addressed.

Displayed Aerodynamic Data

Examples of displayed aerodynamic data using computer graphics are presented. The displays are global views of three-dimensional aerodynamic data. These examples are extracted from different research activities.

Figure 3 shows an over-the-wing view of a contoured nacelle, wing, and fuselage. The streamline paths are derived from linear potential flow calculations using a higher order panel method The geometry, as panelled for the computer code, and the streamlines are displayed.

Figure 4 depicts a duct and the streamlines that visualize the flow field. These streamline paths are also calculated from the same computational code. There are two differences between figures 3 and 4. First, different portions of the streamlines in figure 4 are color mapped according to the relative velocity at the various points along the streamlines. The second difference is that the geometry in figure 4 is translucent to allow some visualization of the stream path behind the geometry.

In figure 5, pressures are displayed on a contoured nacelle, wing, and fuselage. The panels on the surface of the geometry are color mapped according to the magnitude of the isentropic pressure coefficient. The color bar relates the pressure coefficients to different colors. The color bar associated with an image can be changed. This allows the researcher to vary the color bar to inspect pressure ranges of interest with greater detail.

The next two cases deal with data that are obtained from wind tunnel testing. The first is pressure data that was obtained from the 11-ft transonic wind tunnel at NASA Ames by Dryden personnel. In figure 6, the X29A forward-swept wing is shown with color contours. These color contours represent pressure values that are interpolated based on pressures at specific tap locations.

The second case, shown in figures 7 and 8, is a V/STOL concept that was tested at the Air Force Academy. The aircraft and the flow field survey plane are illustrated in figure 7. Figure 8 shows the pressures using color contours. The color contours represent the difference between the local and the free-stream total pressures divided by dynamic pressure.

A future requirement is to visualize experimental aerodynamic data in a three-dimensional display. Current research within the Aircraft Aerodynamics Branch is to develop a method to display contours of wind tunnel pressure data on a complete three-dimensional aircraft geometry that is mathematically defined. The correlation of pressure taps between the mathematical model and the physical wind tunnel model tested is the first phase of the research. The pressure tap locations and values are located on the mathematical model and color mapped. The second phase will involve performing various interpolations of the data across the surface of the geometry between the taps. Figure 9 shows the initial capability that has been obtained. The last phase will display the data as color contours on the mathematically defined surface.

Conclusions

Computer graphics is a capable tool in providing a global and also a detailed local view of three-dimensional aerodynamic data. The images that are generated help to visualize the entire flow field. The use of color provides quick discernment of the data. The future need is to provide a method that displays experimental aerodynamic data in a more effective manner than present methods.

BIBLIOGRAPHY

(1) Carmichael, R. and Gregory, T.: Computer Graphics in Preliminary Aircraft Design. First USA/Japan Computer Conference, American Federation of Information Processing and Information Processing Society of Japan, Tokyo, October 1972, pp. 586-592.

(2) DDM and Dimension III System Reference Manual, CALMA Company, Santa Clara, California, 1981.

(3) Barnhill, R. and Riesenfeld, R.: Computer Aided Geometric Design. Academic Press, 1974.

(4) Clark, J. H.: A Fast Algorithm for Rendering Parametric Surfaces. Proceedings of the Association for Computing Machinery, SIGGRAPH '79 Conference, Computer Graphics Special Issue, August 1979, pp. 7-12.

(5) Newman, W. M. and Sproull, R. F.: Principles of Interactive Computer Graphics. McGraw-Hill Book Company, New York, 1979.

(6) Cozzolongo, J.: Aircraft Geometry Verification With Enhanced Computer Generated Displays. Proceedings of the Thirteenth Congress of the International Council of the Aeronautical Sciences/AIAA Aircraft Systems and Technology Conference, vol. 2, August 1982, pp. 1163A-1163J.

(7) Angell, I.: A Practical Introduction to Computer Graphics. The Macmillian Press, LTD., London, 1982.

(8) Jonas, F. M.: Flowfield Measurements on the Northrop VATOL Concept at High Angles of Attack. Interim Report of NASA-Defense PR A-83011B, USAF Academy, Colorado, July 1982.

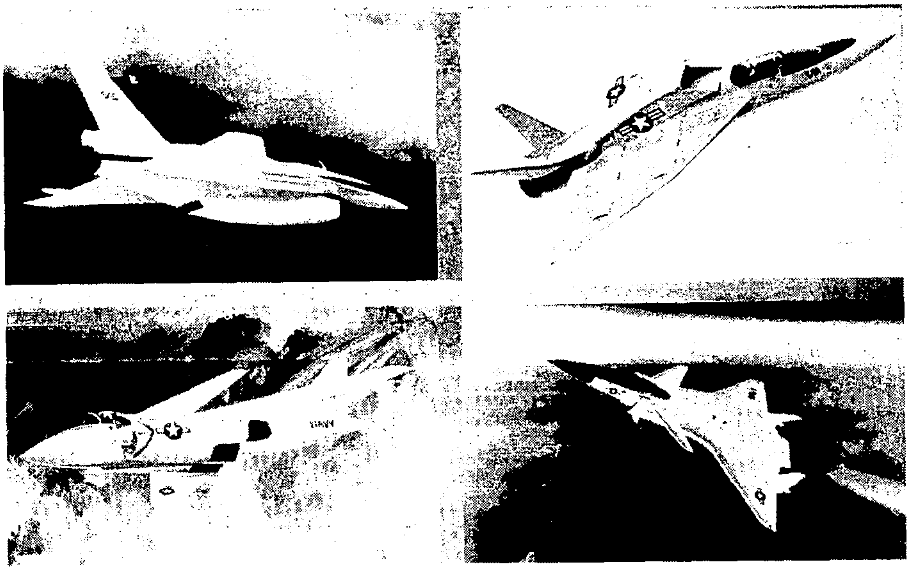


Figure 1.- Single-engine V/STOL fighter/attack aircraft.

Figure 2.- Advanced concept of a contoured turboprop nacelle.

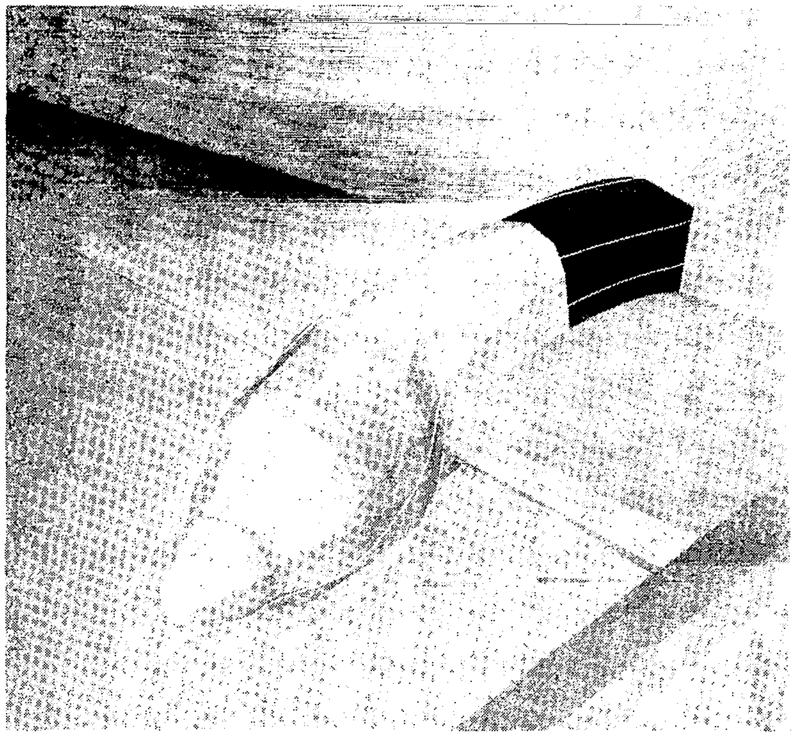


Figure 3.- Calculated streamlines displayed around a nacelle and wing.

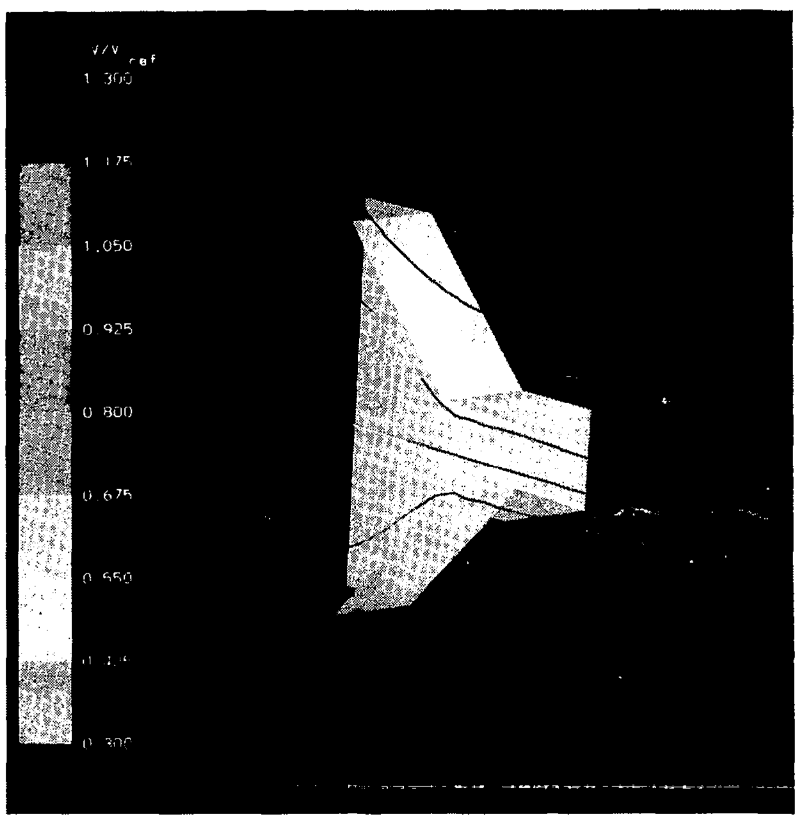


Figure 4.- Streamlines (color mapped to depict relative velocities tracing a flow field through translucent duct).

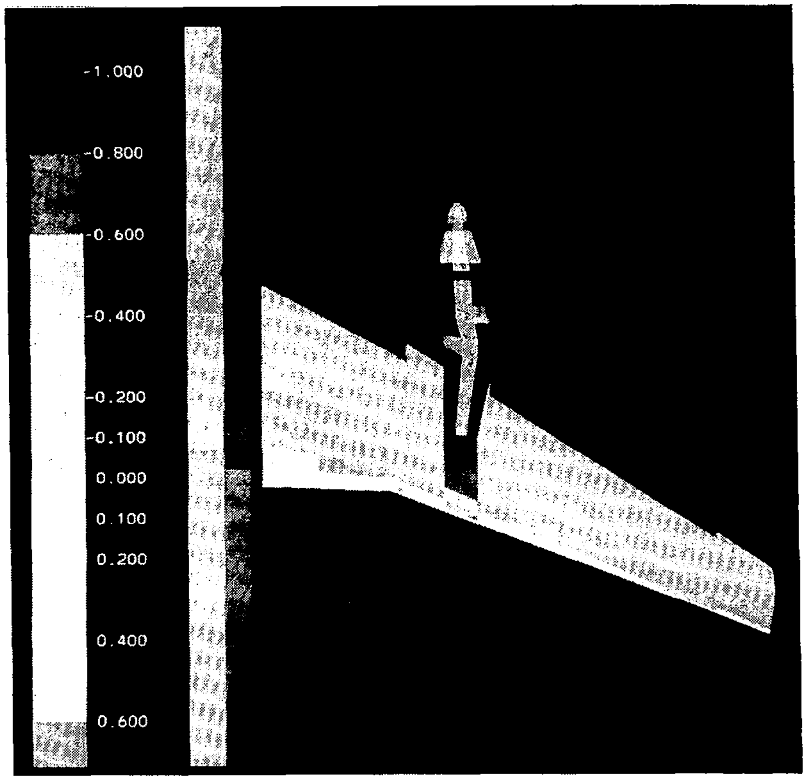


Figure 5.- Pressure coefficients color mapped on a contoured turboprop nacelle and wing.

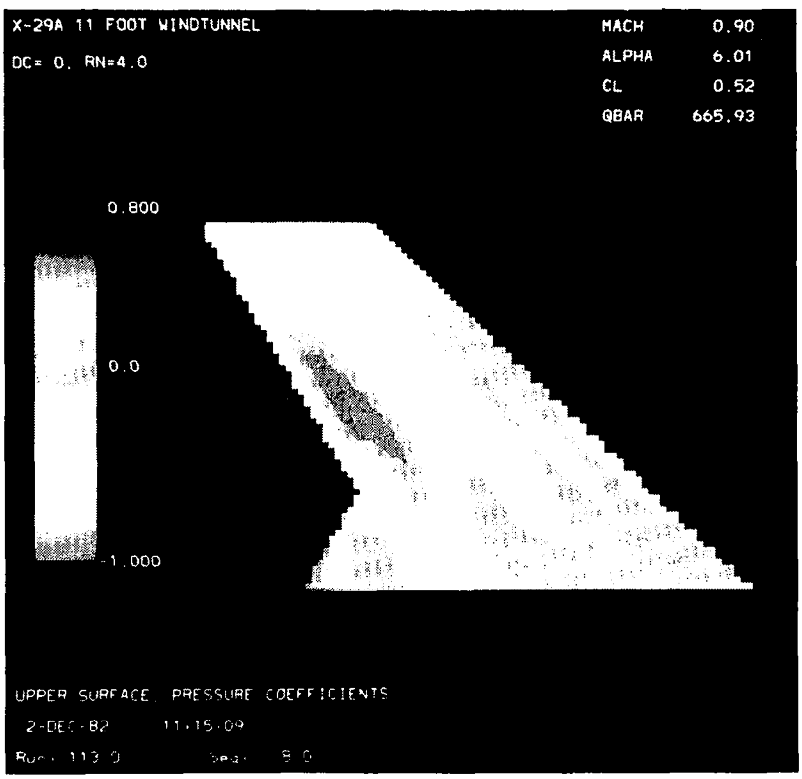


Figure 6.- Color contouring of wind tunnel data on the X-29A forward-swept wing.

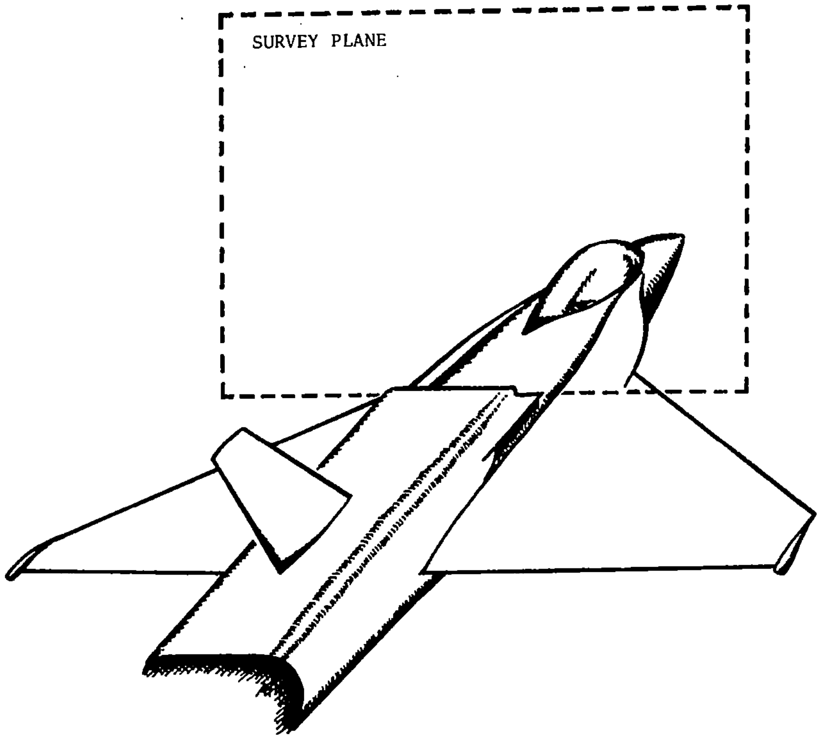


Figure 7.- Aircraft geometry and flow field survey plane used in figure 8.

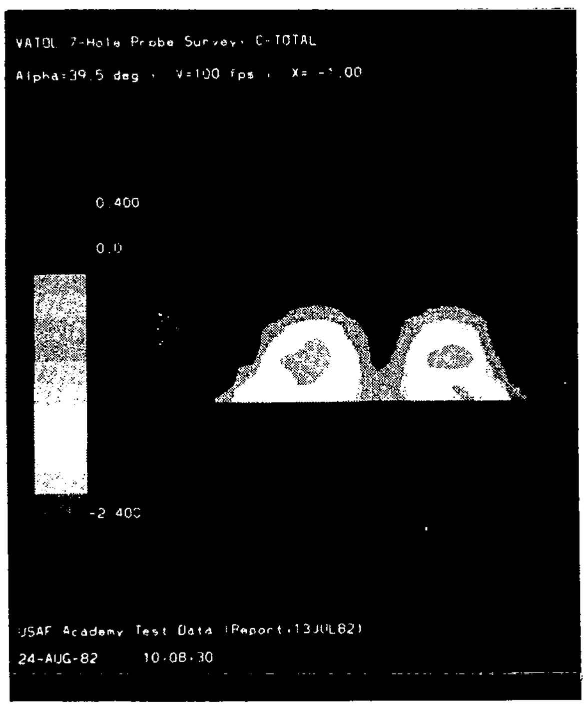


Figure 8.- Color contours representing the difference between local and freestream total pressures divided by dynamic pressure.

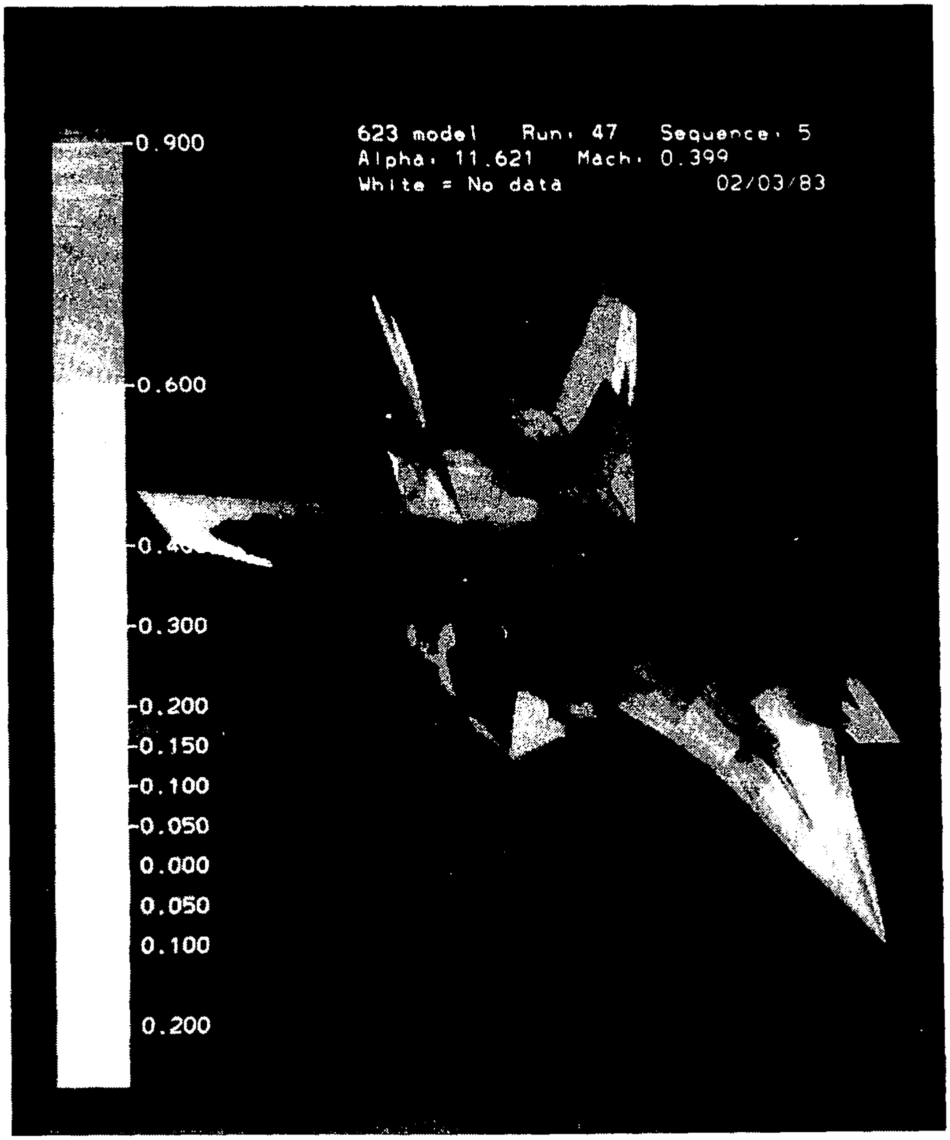


Figure 9.- Initial results at contouring experimental data on a three-dimensional geometry.

INTERACTIVE GRAPHICS FOR GEOMETRY MODELING

Michael J. Wozny
Center for Interactive Computer Graphics
Rensselaer Polytechnic Institute
Troy, New York

The effective generation of complex geometry in modern applications ideally requires a highly interactive graphics system with a capability to render realistic color shaded images. Unfortunately, this combination of display requirements cannot be realized with existing technology. The alternative approach being pursued today and described in this presentation consists of combining the best of both worlds, i.e., an interactive vector capability to create geometry and a raster color shaded rendering capability to sample and verify interim geometric design steps through color "snapshots."

The results described represent the ideas and projects of many graduate students at the RPI Center for Interactive Computer Graphics. Detailed reports identifying specific contributions are included in the references.

The overall research effort at the Center for Interactive Computer Graphics is supported by an Industrial Associates Program consisting of 35 companies and the National Science Foundation. The basic thrust of the research program is to develop the underlying methodology which facilitates computer-aided engineering and design.

INTERACTIVITY OF VECTOR REFRESH WORKSTATIONS

It is well known that vector refresh is the dominant interactive display technology today, and that this interactive capability provides a very effective means for creating 3D geometry. The ability to interactively "move" a 3D geometric image on a graphics display provides needed insight in the creation of complex geometries. Such insight is especially important in solid geometric modeling where the removal of hidden lines is expensive computationally.

Recent 3D systems, such as the Evans and Sutherland PS300, have very high line segment writing speeds which can easily accommodate most current 3D high line density applications. However, interactive applications also need powerful computers for rapid geometric computations. Full 3D commercial geometric design systems require at least 3 to 4 times the computational power of a 2D geometry system. This is evident from benchmark studies done with the Northrop 3D NCAD™ software on an Adage 4370 display/IBM Mainframe system or with the 3D Dassault CATIA™ software on an IBM 3250 display/IBM Mainframe system.

The first project described in this presentation involves the interfacing of the Evans and Sutherland PS300 system directly to an IBM Mainframe channel using the recently announced IBM Device Attachment Control Unit (DACU) [1]. This capability will facilitate the development of a highly interactive graphics capability for geometric modeling. The DACU will also be used to interface into a rich raster environment.

RASTER WORKSTATIONS

Raster display technology continues to penetrate deeply into geometric modeling and CAD/CAM. 1024x1024 pixel 60Hz non-interlaced monochrome raster systems are common today, and provide acceptable line drawing resolution for engineering applications. They are used extensively in CAD/CAM turnkey systems. However, monitor bandwidth is still not wide enough to provide such a non-interlaced capability for color systems. Color raster workstations such as the new Spectrographics system provide good interactive capability for color line drawing on interlaced monitors and have been well received by experienced vector users. However, high resolution non-interlaced raster capabilities are still needed for increased interactivity.

VECTOR-TO-RASTER RENDERING OF SOLIDS AND SURFACES

A shaded color rendering of a mechanical part on a raster display can reveal many important properties not readily apparent on a vector display. Since raster displays cannot duplicate the highly interactive and fast response of a vector refresh system, there is a growing interest in mechanical design to use both display capabilities, by first designing the object on a vector display, scan converting the object into a raster format, and then processing the raster data to obtain a shaded color image.

Capabilities such as the Lexidata Solidview™ system and the Ikonas raster system have hardware capabilities to generate color polyhedral solid models rapidly. The growing 3D surface and solid modeling commercial software packages are accelerating this interest.

This section describes work done at RPI [2] to render geometric vector-generated images on a raster display. The geometric design software used are the IBM experimental solid modeler GDP, and the Northrop 3D NCAD™ geometric system. Both geometric modelers maintain an internal representation consisting of more information than just the wireframe model. In the case of GDP, solid objects are modeled as planar faced polyhedra, while in NCAD™ surface patches are represented as rational bicubic polynomials.

RENDERING GDP SOLID MODELS

The workstation for the GDP modeler is an IBM 3277 GA. The user interacts with menus and commands on an alphanumeric display and the graphics is displayed on a storage tube. With the color extension, a new menu was added to the IBM software which gives the user the ability to assign colors to objects or faces, position the light source, set the background color, etc. [3,4]. Once this is done, the user selects the color output option and obtains a raster shaded image of the wireframe model on either a Raster Technologies Model 1 or a Lexidata 3400 system through an RS 232 connection, at present [2,5]. (This will be replaced with a DACU interface in the near future.)

"Wireframe"-to-raster conversion algorithms can be divided into two phases: (1) the preprocessing phase where the surface information is put into an acceptable format; and (2) the scan conversion phase. The surfaces in GDP are planar polygons which may contain holes interior to the polygons.

The preprocessing phase consists of transforming the polygons into the screen coordinate system, and breaking down each polygon loop into triangles, giving rise to three types of vertices: peak vertex, left side and right side.

A hybrid scan-line and depth buffer procedure is used for scan conversion phase. A one line depth buffer is maintained for the current scan-line with the capability of stacking each pixel to allow the possibility of rendering holes, as long as the holes of a polygon are processed immediately before or after the polygon itself.

Shading interpolation is also implemented to allow smooth shading across facets. To accomplish this, each vertex is assigned a normal vector which is the average of all the normals to polygons sharing the vertex. (It is therefore important that the geometric modeler flag faces which belong to a curved surface.) Pixels interior to the polygon are colored by linearly interpolating the normals at the vertices, giving the user the option of either Gouraud or Phong shading. [6]

RENDERING NCAD™ RATIONAL BICUBIC SURFACE MODELS [2]

The NCAD workstation consists of a high performance vector refresh display (IBM 3250 or Adage 4370 etc...). The raster interface allows the user to color surfaces and background, and set the direction and intensity of the light source(s). This data is passed to a separate process via a dataset, allowing the model to be colored and shaded without tying up a workstation.

The preprocessing phase of the "wireframe"-to-raster algorithm consists of first transforming each patch into the screen coordinate system,then discretizing it into planar facets. The method used to facet a patch is analogous to the Lane-Riesenfeld algorithm [7] except that a rational parameterization is involved. The surface is divided into sub-patches until each sub-patch is planar to within a specified tolerance, usually a few pixels. To subdivide a patch the coefficients are first converted to the Bernstein basis. The most complex operation in the division process is division by multiples of 2. Consequently the method is attractive for implementation in hardware. The use of the Bernstein basis also greatly simplifies the computation of the normal vector at the corners of the sub-patch. This is given as the cross product of the two vertices of the characteristic polyhedron adjacent to the corner vertex.

After a sub-patch has been classified as planar it is triangulated for the next phase. Care must also be taken to pass on the triangles which lie on 'cracks' between patches subdivided to different levels.

The scan-line algorithm provides anti-aliasing up to a resolution of 4K by 4K. To accomplish this degree of anti-aliasing and still be fast, a look-up technique is used in conjunction with the Bresenham algorithm. That is, the program walks along the edge of the triangles as though it were displaying a vector on a 4K by 4K display. However, only 512 scan-lines are computed. When an edge crosses the edge of a real pixel (512 x 512) the edge crossings are noted and the area covering the real pixel is computed by a table look-up. This algorithm is somewhat less complex than that used for solid models since no holes are processed.

Although color shaded renderings of a solid or surface significantly enhance certain engineering applications, both implementations described above still take a long time to compute and are not considered interactive.

RENDERING SUPERQUADRIC SOLID MODELS

Superquadric objects extend the traditional quadric geometric primitives into a family of flexible forms. A superquadric ellipsoid is generated by a spherical cross product of two parameterized superellipses [8]. The two parameters (one for each superellipse) which define the "squareness" of each superellipse collectively represent a two-parameter family of 3D surfaces. The carteasian coordinate representation of a superquadric surface allows the definition of an inside-outside function, and is consequently, a representation of a solid. The volume of certain convex superellipsoids can be expressed as a ratio of two gamma functions [9].

The cross product of the two tangent vectors through a common point along the two generating superellipses defines the normal to the surface at that point. This normal is easily calculated and provides a direct method for shading curved objects. Rendering of superquadrics has been accomplished using a fast ray-tracing algorithm for creating high quality, anti-ailiased color shaded images [10,11]. Ray-tracing has also been used to perform boolean subtraction of superquadric primitives. Current activity involves the development of an easy to use, interactive wireframe interface for creating superquadric images [12].

CONCLUSION

At present, raster systems cannot match the interactivity and line-drawing capability of refresh vector systems. Consequently, an intermediate step in mechanical design is to create objects interactively on the vector display and then scan convert the wireframe model to render it as a color shaded object on a raster display. Several algorithms were presented for rendering such objects using the IBM SOLID experimental software GDP and the Northrop 3D NCAD commercial software. Superquadric solid primitives extend the class of primitives normally used in solid modelers.

New work involves the Dassault Systems CATIA™ software.

REFERENCES

1. "Device Attachment Control Unit", Reference and Operation Manual, IBM, Poughkeepsie, NY, DACU-ROM-0.

2. Sabella, Paolo, Project Reviews, Center for Interactive Computer Graphics, RPI, Troy, New York, May, August, and December 1982.

3. Puccio, Phil, "Color Display Support for the Geometric Design Processor", User's Manual and Technical Report, CICG, RPI, 1981.

4. Zawada, John, "Color Mixing Program", Project Report, Center for Interactive Computer Graphics, RPI, Troy, New York, May 1982.

5. Snyder, Derek, "A Color Raster Addition to the GDP System with Full Hidden Surface Removal", Master's Thesis and Technical Report, CICG, RPI, 1981.

6. Foley, J., and Van Dam, A., "Fundamentals of Interactive Computer Graphics", Addison-Wesley Publ. Co., Reading, MA, 1982.

7. Lane, J., and Riesenfeld, R., "A Theoretical Development for the Computer Generation of Piecewise Polynomial Surfaces", IEEE Trans., PAMI 2, 1980, pp. 35-46.

8. Barr, Alan H., "Superquadrics and Angle-Preserving Transformations", IEEE Computer Graphics and Applications, January 1981.

9. Barr, Alan, Project Review, Center for Interactive Computer Graphics, RPI, Troy, NY, May 1982.

10. Barr, Alan, and Edwards, Bruce, Project Review, Center for Interactive Computer Graphics, RPI, Troy, New York, May, August and December, 1981.

11. Edwards, Bruce, "Implementation of a Ray-Tracing Algorithm for Rendering Superquadric Solids", Master's Thesis, RPI, December, 1982.

12. Breen, David, Project Review, Center for Interactive Computer Graphics, RPI, Troy, New York, December 1982.

INTERACTIVE COMPUTER GRAPHIC SURFACE MODELING OF THREE-DIMENSIONAL SOLID DOMAINS FOR BOUNDARY ELEMENT ANALYSIS

Renato Perucchio * and Anthony R. Ingraffea **

Department of Structural Engineering, Cornell University
Ithaca, New York

Recent years have seen the establishment of the boundary element method (BEM) as a valid tool for solving problems in structural mechanics and in other fields of applied physics (ref. 1). The authors are currently working on the development of an integrated interactive computer graphic system for the application of the BEM to three-dimensional problems in elastostatics. The integration of interactive computer graphic techniques and the BEM takes place at the preprocessing and postprocessing stages of the analysis process, when, respectively, the data base is generated and the results are interpreted. This paper will focus on the interactive computer graphic modeling techniques used for generating and discretizing the boundary surfaces of a solid domain.

The necessity of interactive model generation for three-dimensional BEM analysis stems from the experience acquired with the application of the finite element method to the same class of problems. It has been shown that the high cost of performing FEM analysis on three-dimensional domains is due largely to the effort required in defining and checking the geometrical data, element topology, boundary conditions, material properties, and loadings. Fewer difficulties are encountered in creating a geometrical model and related data base for BEM analysis of a three-dimensional domain, since the BEM requires, in general, only a discretization of the boundary surfaces, thus eliminating the necessity of partitioning the inside of the domain. The absence of internal nodes represents a significant relaxation of constraints on the mapping algorithms that are usually applied in the automatic generation of discrete meshes. Now nodes and topologies can be defined independently on each boundary surface, and consistency requirements are limited to the nodes on the contours of adjacent surfaces.

Despite these advantages, the task of inputting and discretizing a family of three-dimensional surfaces for BEM analysis will remain long and tedious unless a better interaction between analyst and machine is introduced to enhance the creative aspects of designing and modelling. From this point of view, interactive computer graphic techniques offer a highly efficient way of generating three-dimensional surfaces with the minimum amount of direct digital input. The present interactive graphic code is written in FORTRAN-77 and runs on a Digital Equipment Corporation VAX 11/780 and an Evans and Sutherland Picture System 2 high-speed refresh vector display. Data are fed to the machine by a pen and digitizing tablet and are immediately displayed on the vector scope. Since the BEM requires the discretization of the domain surface, the geometrical model consists of a family of discretized, generally curved, surfaces that the analyst will input using the digitizing tablet as much as possible. However, both digitizing tablet and display scope are intrinsically two-dimensional devices and therefore ill suited for handling three-dimensional geometries. Consequently, an appropriate strategy must be devised to permit input of geometrical and topological data without extensive use of a keyboard to type in nodal-related information.

---

* Graduate Research Assistant, Department of Structural Engineering and Program of Computer Graphics.

** Associate Professor of Structural Engineering.

The approach followed here is based on the use of mapping algorithms that allow the discretization of a three-dimensional surface based on interpolating between boundary and cross-sectional curves (ref. 2). The assumption is made that boundary and cross-sectional curves are planar and therefore easily traceable with the digitizing tablet. This is equivalent to the procedure used by the writers to generate solid, three-dimensional meshes for FEM analysis by discretizing cross sections into planar meshes and then using blending functions to interpolate between the sections (ref. 3). An important difference in the present case is that only the first and the last cross sections need to be meshed. The interior cross sections are simply discretized, planar contours. The problem of generating a family of three-dimensional general surfaces that bound a solid domain is reduced to the simpler one of creating planar contours and two-dimensional meshes for which computer graphic techniques have proved extremely effective.

The analyst controls the surface discretization process by selecting and positioning the cross sections and by grading the relative size of the lofted surface elements. After completing the lofting procedure, he can examine separately one surface at a time using the dynamic viewing capabilities and depth cueing built into the high-speed refresh vector display.

The generation of the geometrical model and of the boundary value data for a turbine blade assembly under distributed surface load will provide an example to illustrate the flexibility and the capability of the interactive preprocessor. The complete geometrical model (shown in figure 1) consists of two parts, one corresponding to the root of the blade, the other to the blade proper. The sharp variation of cross-sectional geometry between the root and the aerodynamic surface suggests a natural way of dividing the complete domain into two subdomains to be generated separately. The cross section indicated as "interface" in figure 1 represents the surface shared by the two subdomains and needs to be discretized only if domain substructuring is going to be used at the analysis level. In this case, the unknowns in each subdomain are condensed out in terms of the unknowns of the nodes belonging to the interface. Therefore, this cross section requires discretization even though it does not belong to the family of boundary surfaces which enclose the original domain.

Cross-Sectional Data - A module of an existing interactive two-dimensional FEM preprocessor (ref. 4) is used to input the outlines and generate the planar meshes in the x-y plane. Each curve is input by a digitizing tablet either by tracing it or by first pointing at key points on a grid and then using curve generating routines. The user also selects the number of nodes and their spacing on each curve. Where required, a planar mesh is generated by first subdividing the outline into simpler subregions with 2, 3, or 4 edges and then partitioning each subregion with triangular or quadrilateral elements by transfinite mapping. Figure 2 shows one of the cross sections used for lofting the first subdomain.

Mesh Lofting - The next task is generation of lofted surfaces. Once each cross section is completely defined, it is ready to be positioned in a three-dimensional coordinate system. This transformation is accomplished in one of two alternative ways according to whether or not the section is to be normal to a reference axis. In the first case, the user must indicate the normal axis, the section scale factor, the angle of rotation about the normal axis, and the final x, y, z coordinates of the sectional axes origin. Alternatively, the user selects three nodes on the cross-sectional mesh and enters their x, y, z coordinates; by the correspondence of planar and three-dimensional locations of the nodes, the necessary information for transforming the entire cross section is generated.

After positioning all the necessary cross sections, the user specifies the type of surface elements, the number of nodes in the lofting direction, and the relative spacing between the lofted nodes. The lofting algorithm then generates the surfaces. For the first subdomain, figure 3 shows some of the cross sections already positioned in the three-dimensional Cartesian space. Figure 4 depicts the complete three-dimensional mesh. To reduce the complexity of the display, the surface elements may be removed, and only the boundary curves of each facet are then displayed (as in figure 5). It must be emphasized, however, that dynamic viewing capabilities and depth cueing of the refresh vector display, especially the ability to view slow continuous rotation, allow the user to perceive even the entire mesh in its complete three-dimensionality without making use of time consuming hidden-line removal algorithms.

The second subdomain requires at least three cross sections to model the twisting and reduction in scale of the airfoil along the length of the blade. Figure 6 shows the cross sections already positioned and one of the lofted facets meshed by selecting 10 partitions in the lofting direction with a ratio in length of 2:1 between the first and the last band of elements. The mesh for the complete blade-root assembly is presented in figure 7.

A further example of the preprocessor modelling capability is presented in figure 8, which shows the three-dimensional model of a concrete dam buttress complete with foundation. The surface mesh has been assembled by lofting separately the foundation, the core of the dam buttress, and the upper slab. The intersection between the buttress and the foundation is illustrated in figure 9. The entire model requires the creation and positioning of only four main cross sections.

REFERENCES

1. Banerjee, P. K., and Butterfield, R., Boundary Element Methods in Engineering Science, McGraw-Hill Book Company (UK), London, 1981.

2. Gordon, W. J., and Hall, C. A., "Construction of curvilinear co-ordinate systems and applications to mesh generation", International Journal for Numerical Methods in Engineering, vol. 7, no. 4, 1973, pp. 461-477.

3. Perucchio, R., Ingraffea, A. R., and Abel, J. F., "Interactive computer graphic preprocessing for three-dimensional finite element analysis", International Journal for Numerical Methods in Engineering, vol. 18, no. 6, June 1982, pp. 909-926.

4. Han, T. Y., "A general two-dimensional, interactive graphical finite/boundary element preprocessor for a virtual storage environment", M. S. Thesis, Department of Structural Engineering, Cornell University, 1981.

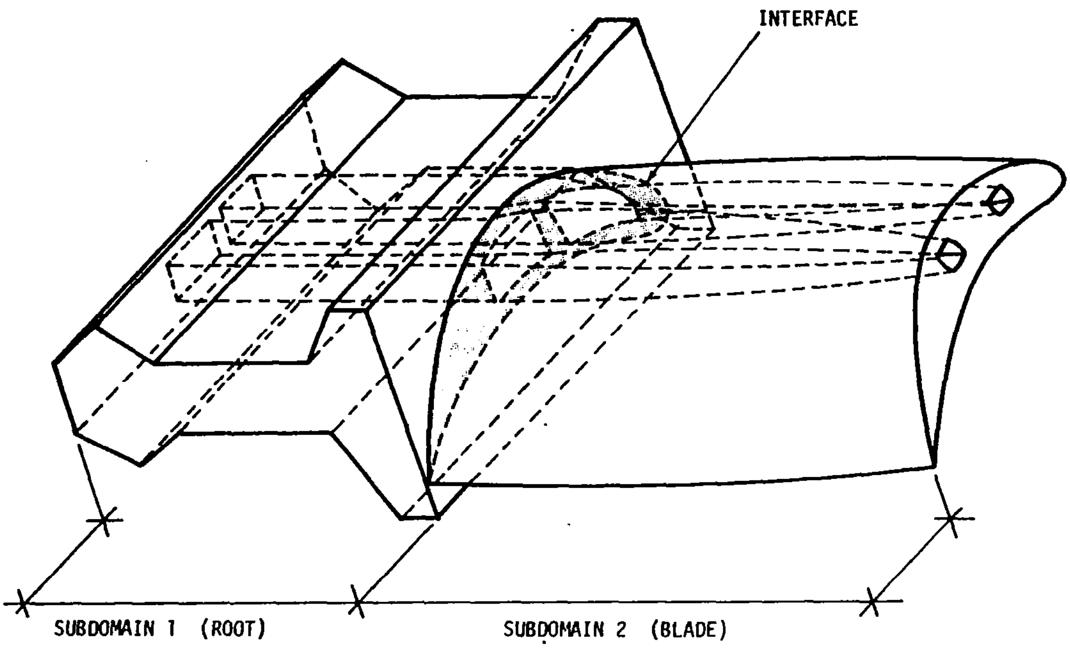


Figure 1.- Geometry of turbine blade assembly used in example problem.

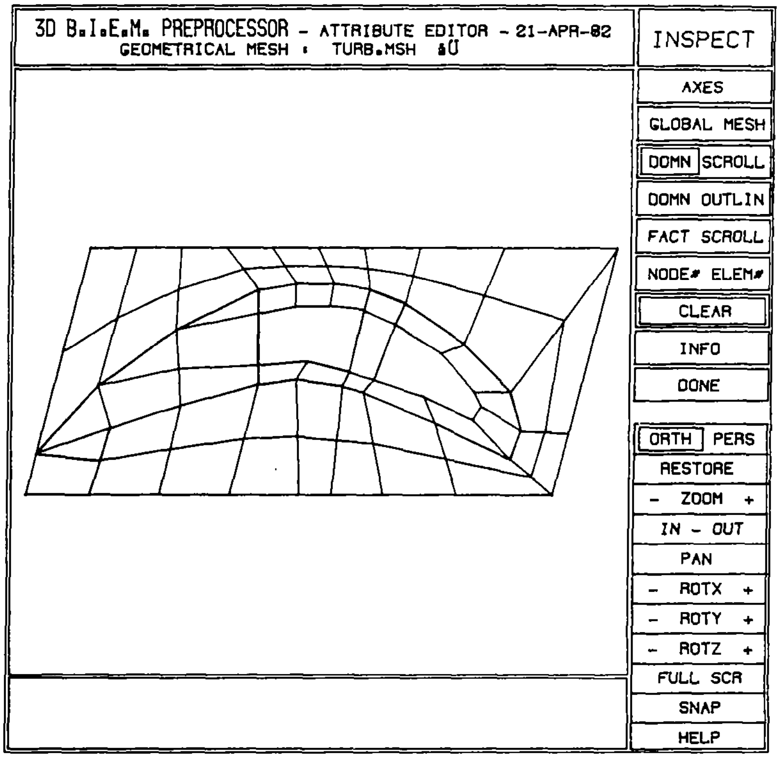


Figure 2.- Cross section used for lofting subdomain 1.

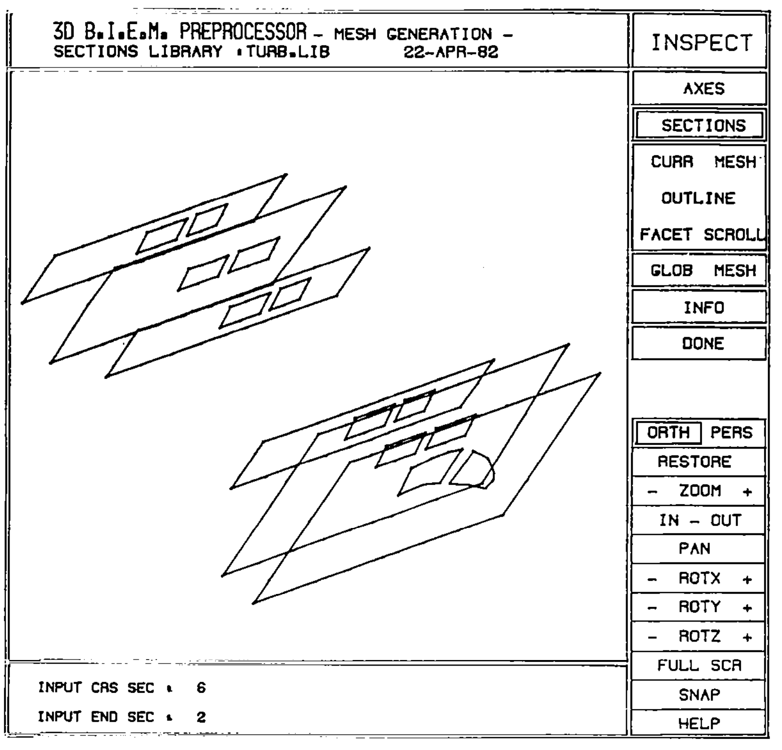


Figure 3.- Positioning of cross sections for subdomain 1.

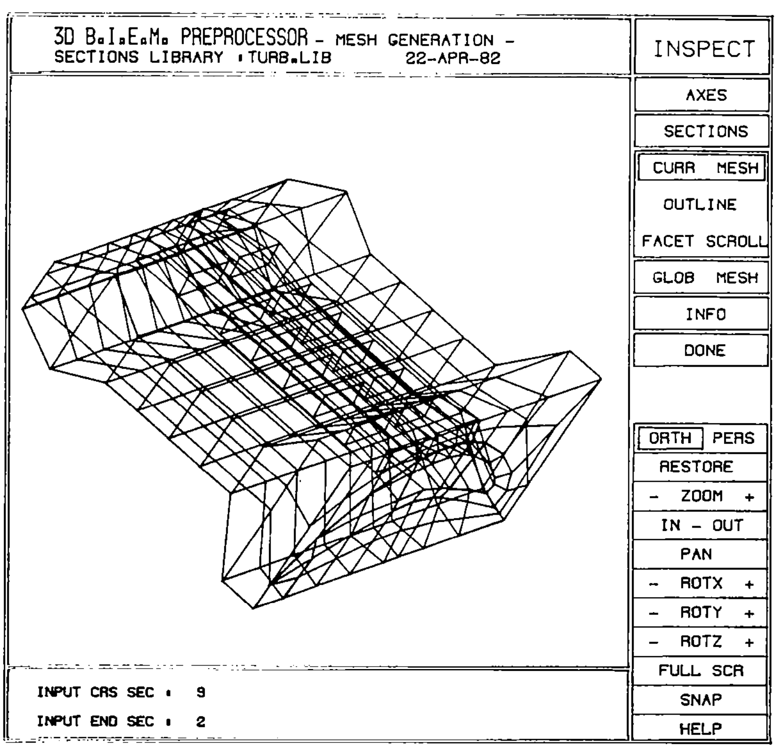


Figure 4.- Geometrical model for subdomain 1.

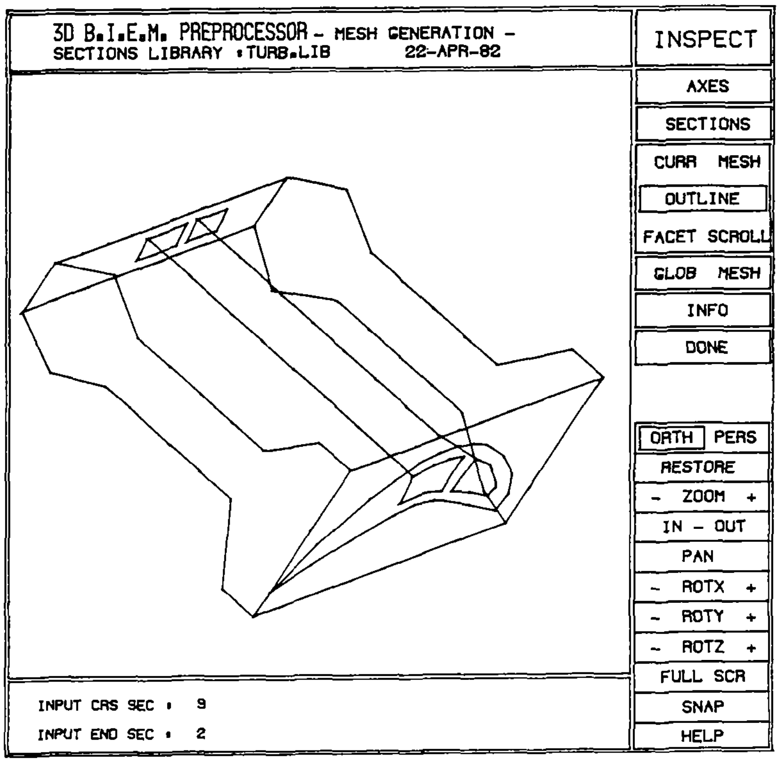


Figure 5.- Boundary curves for subdomain 1.

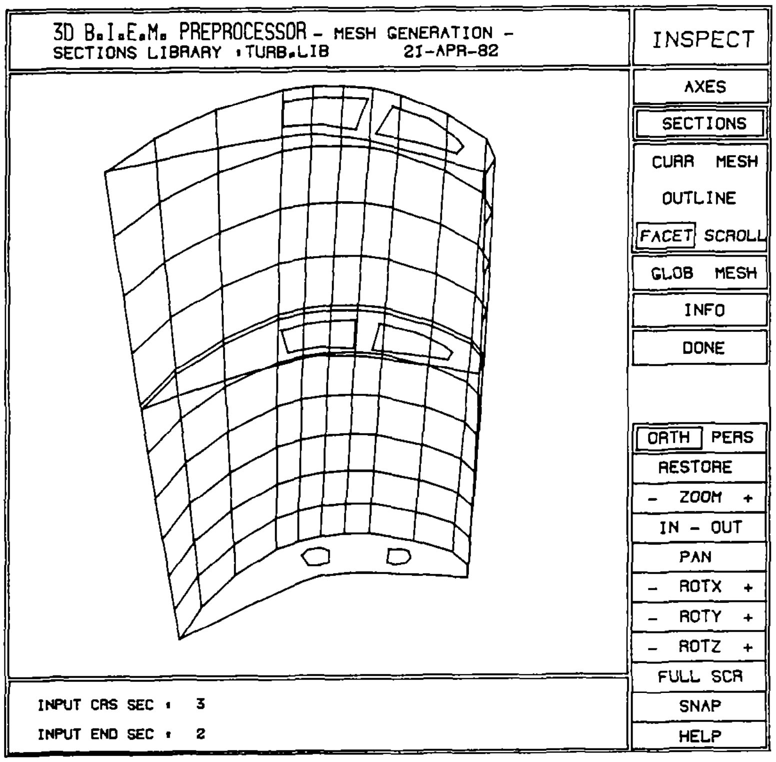


Figure 6.- Cross sections and lofted facet for subdomain 2.

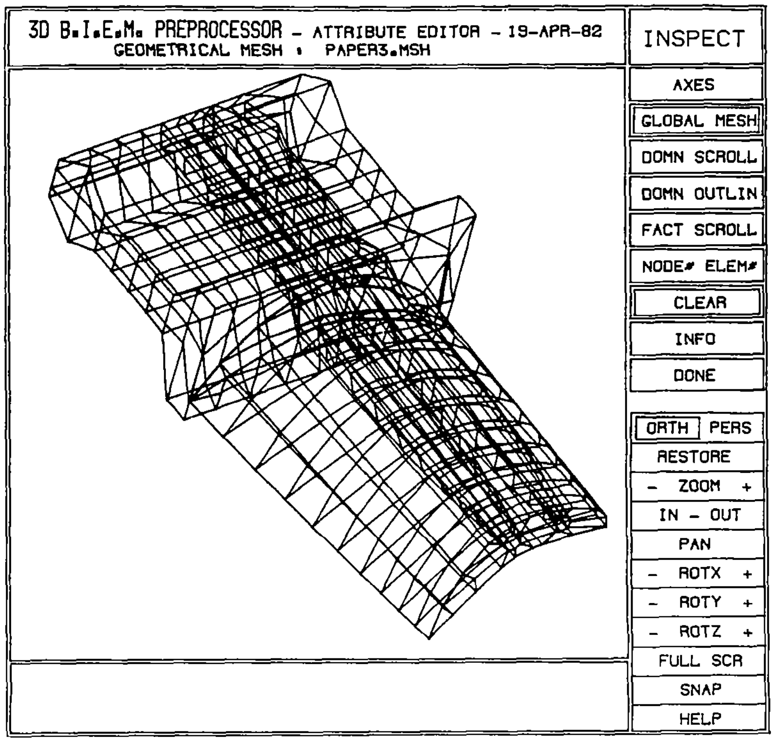


Figure 7.- Complete geometrical model.

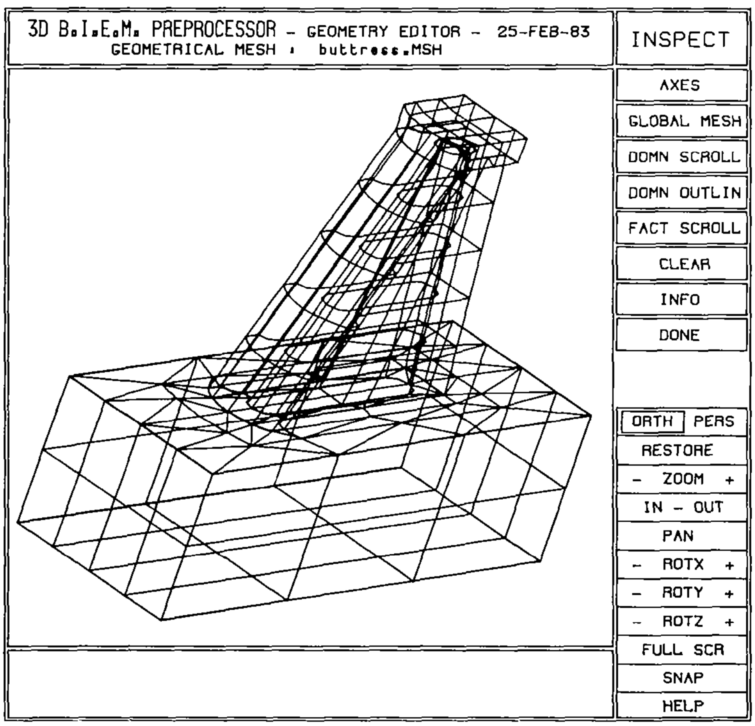


Figure 8.- Geometrical model of buttress dam with foundation.

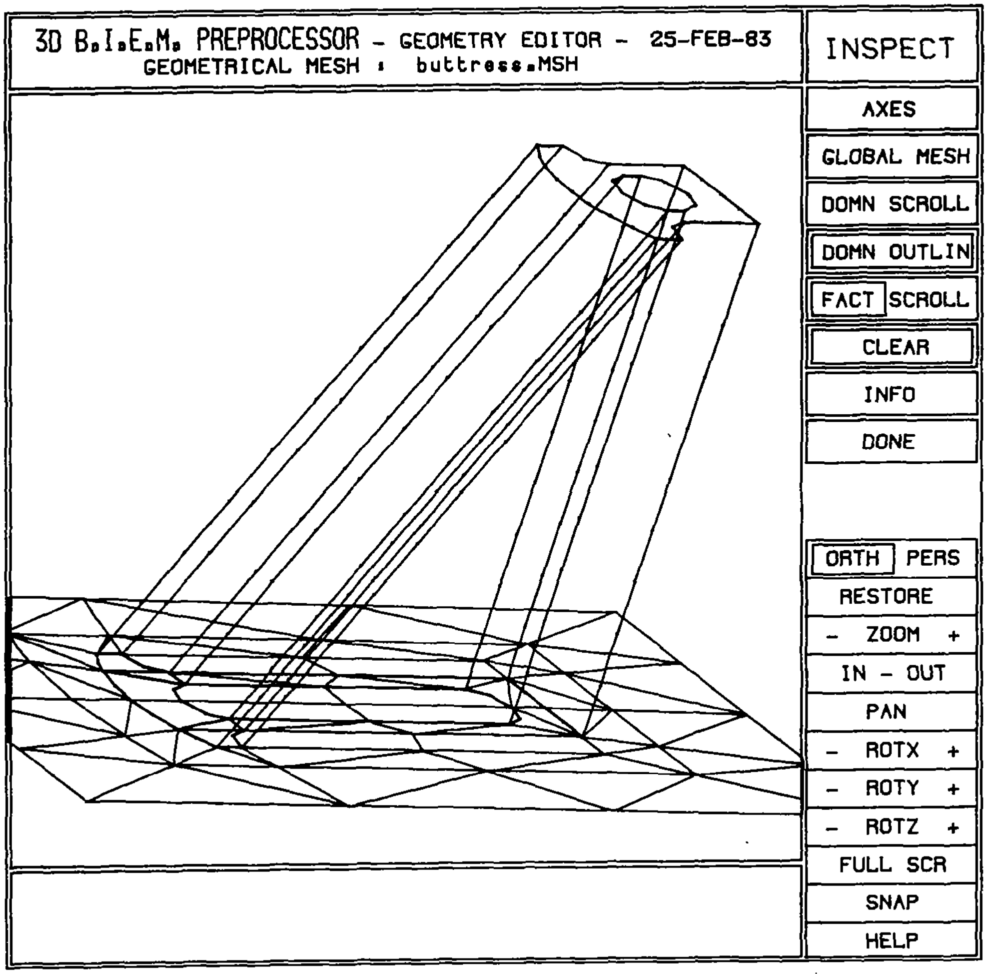


Figure 9.- Boundary curves of buttress dam model.

INTERACTIVE WIRE-FRAME SHIP HULLFORM GENERATION AND DISPLAY

D. E. Calkins
Ocean Engineering Program
Department of Mechanical Engineering

J. L. Garbini
Department of Mechanical Engineering

J. Ishimaru
Department of Applied Mathematics

University of Washington, Seattle, Washington

An interactive automated procedure for generating a wire-frame graphic image of a ship hullform, which uses a digitizing tablet in conjunction with the hullform lines drawing, has been developed. The geometric image created is displayed on an Evans & Sutherland PS-300 graphics terminal for real time interactive viewing or is output as hard copy on an inexpensive dot matrix printer.

The geometry of a ship's hull is described on a drawing known as the "lines drawing." A set of lines consists of three plans or views, which include the elevation or vessel profile (sheer plan), a view looking down on the vessel (half-breadth plan) and a set of transverse sections (body plan), Fig. 1. Each of the three plans on the lines drawing represents a reference plane on which have been projected the various lines and points of the hullform. These three projections allow the naval architect to determine the relative positions in space of all points and lines on the ship.

While mathematical ship hull modeling concepts are being developed in which the hullform is mathematically described by a data base which may be used for additional computations such as hydrostatics and hydrodynamics, Ref. 1, it is frequently desired to view only a three-dimensional geometric rendition of the hullform. This allows the naval architect to check the fairness of the lines or to use the geometric image for special applications such as coupling with a time domain dynamics program for animated rigid body motions display.

The Evans & Sutherland PS-300 Interactive Graphics System, which is based on a monochromatic vector refresh terminal, Fig. 2, provides the user with the ability to model complex two- and three-dimensional objects and to specify viewing parameters and geometric transformations which affect these objects. With the aid of a wide range of "black box" functions, the user is able to construct networks which allow interactive control over all aspects of the display of data, as well as the execution of various arithmetic and boolean operations which facilitate real time motion of objects and logical branching based upon user inputs.

The PS-300 allows the viewing of an image with no hidden line removal, but with real time dynamic viewing and depth cueing. It operates in a distributed graphics environment; that is, non-graphical tasks are separated from graphical tasks, and the latter are performed locally by the PS-300 via its graphics control processor (GCP). The GCP consists of an M68000 micro-processor and is dedicated to handling graphics tasks and guiding the interface between the host and the PS-300. The PS-300 acts as a self-sufficient graphics display system, while the host (in this case, a DEC PDP 11/44) transmits instructions via string commands in a command file, acts as a permanent storage facility for those files, and is used as a receptacle for output data from the PS-300. User interaction with the graphics display may be accomplished via any of three main devices: (1) the keyboard, which allows direct operator input of PS-300 commands and contains 36 user-programmable function keys which can be used to select display options, (2) the digitizing tablet with stylus, which can be used for point digitizing, menu selection, and special features such as "rubber banding," and (3) the control dials unit, which consists of eight dials which can be defined as part of a network to provide user input of scaling factors, rotations, translations, etc.

The interactive automated procedure has been developed based on the utilization of existing ship hullform lines drawings and requires no programming on the part of the user. It should be emphasized that a mathematical data base for the hullform is <u>not</u> developed by this procedure. The procedure is especially appropriate for hullforms known as hard chine hulls, Fig. 3, wherein hull transverse curvature is very slight and essentially straight lines are used to generate the transverse sections of the body plan.

The lines drawing is placed on a large digitizing tablet, in this case a 3' x 4' Tektronix 4954 board, Fig. 4, and the desired nodal points on each transverse section are digitized to develop the station section shape. A vector list of the nodal points is thus generated for each transverse section along the length of the hull. The procedure is not particularly appropriate for ship hullforms with large amounts of transverse curvature, unless the user is willing to digitize a large number of points on each station.

After a pre-selected number of transverse sections have been digitized from the body plan, the transverse sections image is then displayed on the terminal, Fig. 5, and the longitudinal lines are generated by connecting appropriate nodal points using the "rubber banding" feature of the PS-300, Fig. 6. At this point, the hullform vector list is extended, and the translation, rotation and scaling functions are added by a template. The procedure is depicted in flow chart form in Fig. 7.

A boat geometry which was used as an example of the procedure, known as HYCAT for HYdrofoil CATamaran, is shown in Fig. 8. HYCAT combines a planing catamaran hull with two fully submerged hydrofoils mounted transversely in tandem fore and aft.

Three coordinate systems are used in the image generation process. An initial global coordinate system (X_G, Y_G, Z_G) is used for the basic hullform, with the Y_G axis coincidental with the bow of the ship, the X_G axis along the keel of the ship and the Z_G axis to port, Fig. 9. A similarly oriented coordinate system is used for each specific item that is added to the hullform to create a complete ship geometry. For instance, in the case of the demonstration hullform, the hydrofoils (X_f,Y_f,Z_f) and the rudders (X_r,Y_r,Z_r) were developed in their own systems and then added to the global system. Finally, a coordinate system (X_B,Y_B,Z_B) with its origin at the ship center of mass, with the X_B axis forward, the Y_B axis upward, and the Z_B axis to starboard, is used to define the rotational degrees of freedom as viewed on the graphics terminal. The center of mass of the ship geometry was chosen as the origin of this system since it is intended to use the graphics image in conjunction with a rigid body time domain simulation for animated motions display, where the equations of motion are written with respect to that particular coordinate system.

A program which generates the vector list from the digitizing tablet has been written. This vector list is then transformed into the PS-300 command language by a second program. Additional programs have been written which then create (attach) the necessary commands for viewing the hullform image on the PS-300. The PS-300 function network and data structures are depicted in Fig. 10. This includes specific functions for the keys on the keyboard, such as the addition or deletion of image components. The dial functions include three translations in the X, Y and Z directions in a screen oriented world coordinate system and three rotations about the X_G, Y_G and Z_G axes.

The boat geometry ensemble presently comprises seven vector lists, including the hull, or the portion from the keels up to the sheer line, the superstructure in four sections (pilothouse, forward, center and aft), and two individual hydrofoils and rudders. A composite ship geometry may be created by adding or deleting any one of these components through the use of the keyboard function keys. Approximately 3500 vectors comprise the total geometry. The image may be viewed in either orthographic-axonometric, Fig. 11, or perspective projections, Fig. 12, including depth cueing for image enhancement. Hard copy is obtained directly from the PS-300 terminal either through 35 mm or polaroid photographs, or video tape.

In addition, a dot matrix printer is used as an inexpensive means of producing hardcopy drawings of both the major orthogonal views, Fig. 13, and the orthographic-axonometric views, Fig. 14. These images are derived directly from the model data base in the host computer, rather than as an output of the graphics system. The vector lists comprising the model are rotated and then projected onto the desired viewing plane. All vectors in the final coordinate system are converted to a bit map (512 x 390) which is printed, in raster fashion, with eight lines in each pass. Although hidden lines are not removed, depth cueing is achieved by adjusting the dot density (roughly equivalent to the line darkness) as a function of the distance from the viewing plane.

An advantage of the bit-mapped hardcopy is that the speed of reproduction is relatively insensitive to the number of vectors in the image. A drawing of arbitrary complexity is printed in approximately 20 seconds.

REFERENCE

1. "Computer Aided Hull Design--Computer Applications in the Automation of Shipyard Operation and Ship Design," IV, IFIP/IFAC Fourth International Conference, U.S. Naval Academy, Annapolis, MD, 7-10 June 1982, pp. 155-205.

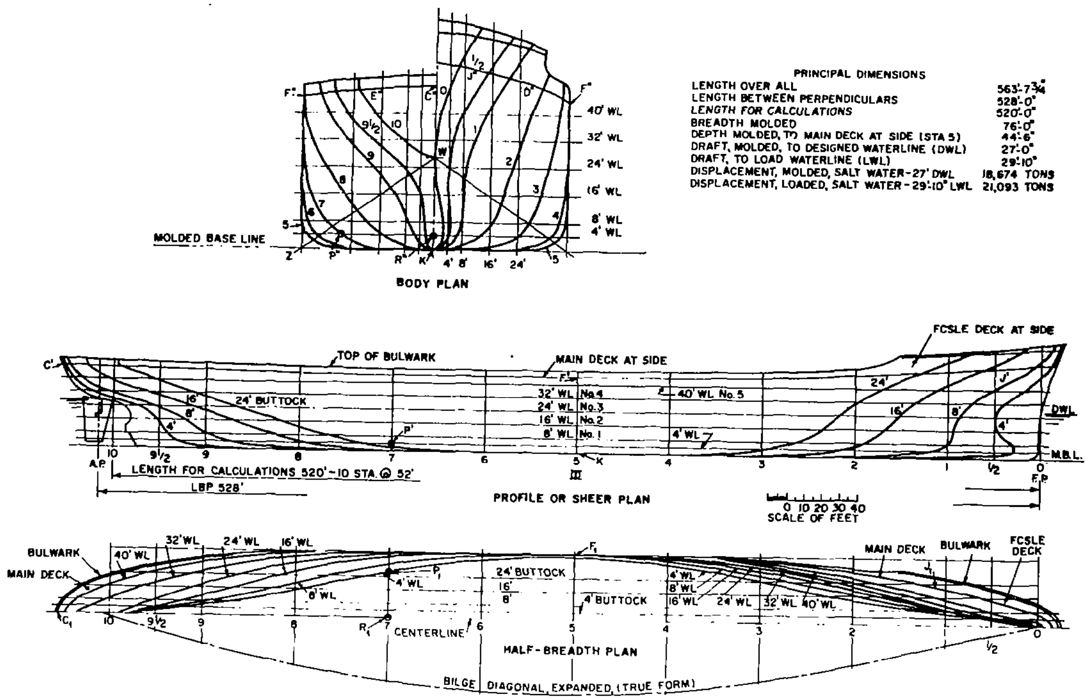


Figure 1. Ship lines drawing.

Figure 2. Evans & Sutherland PS-300 graphics system.

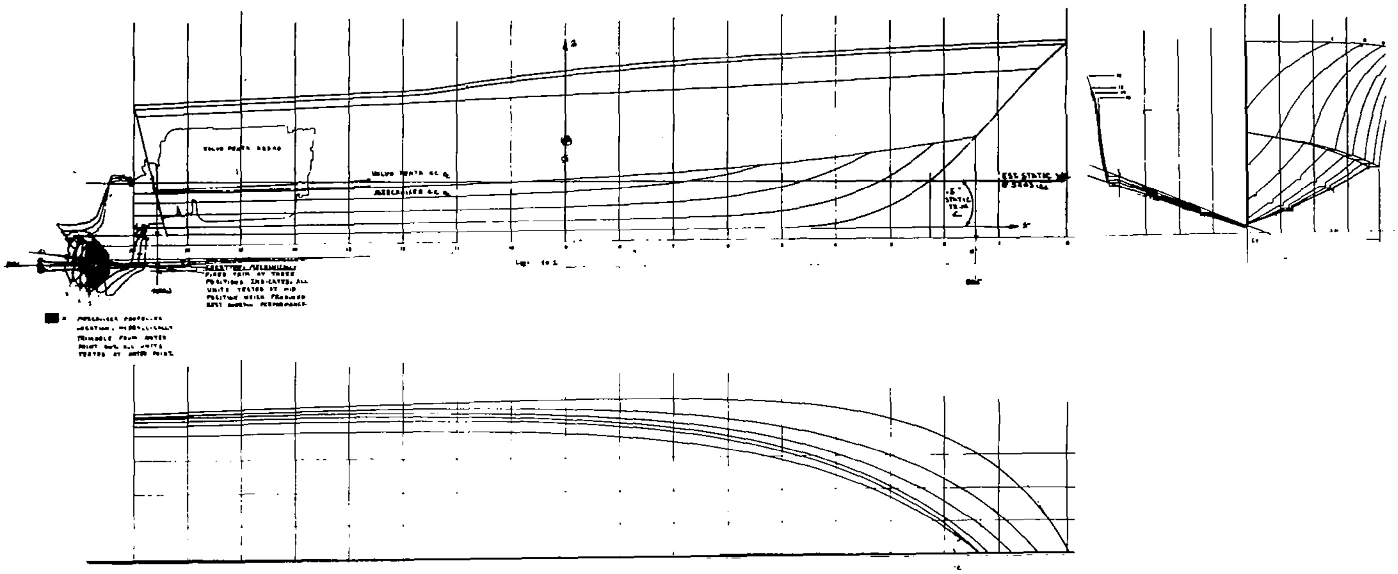


Figure 3. Hard chine hull lines drawing.

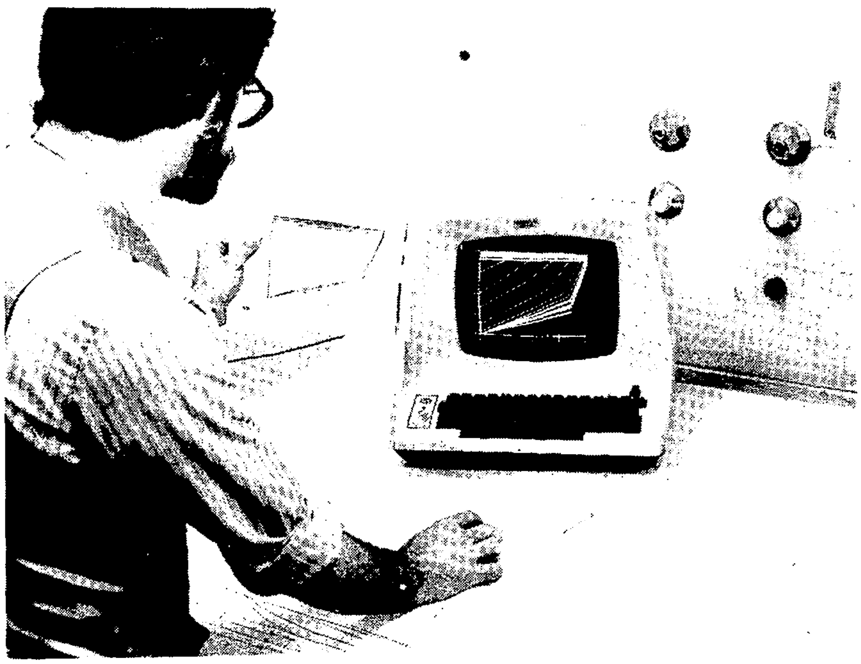


Figure 4. Digitizing body plan.

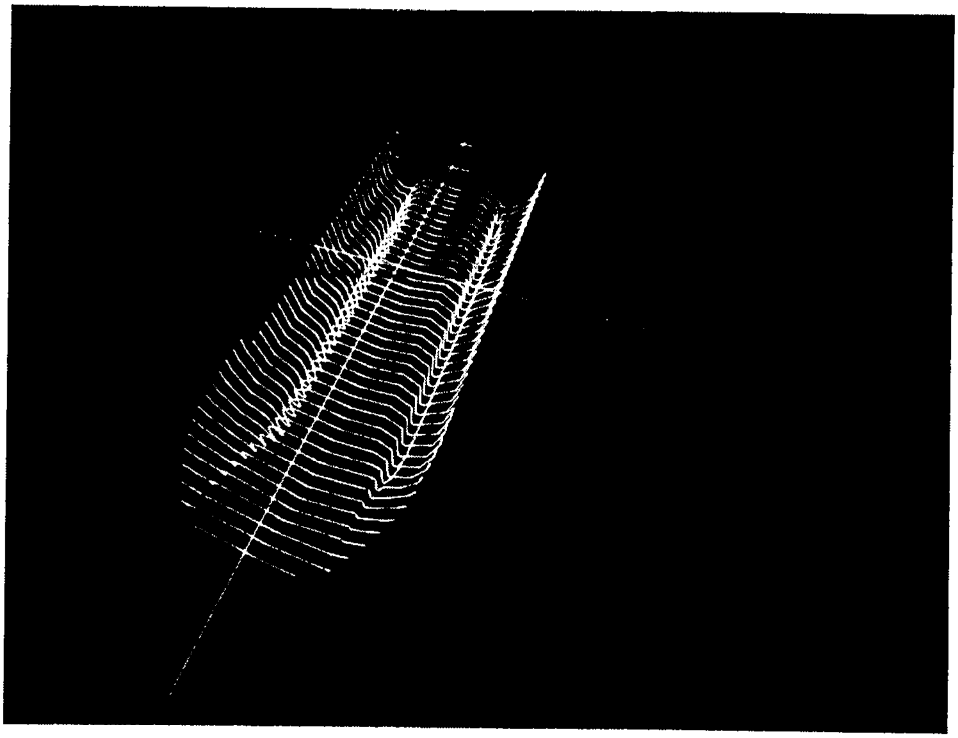


Figure 5. Hull transverse sections.

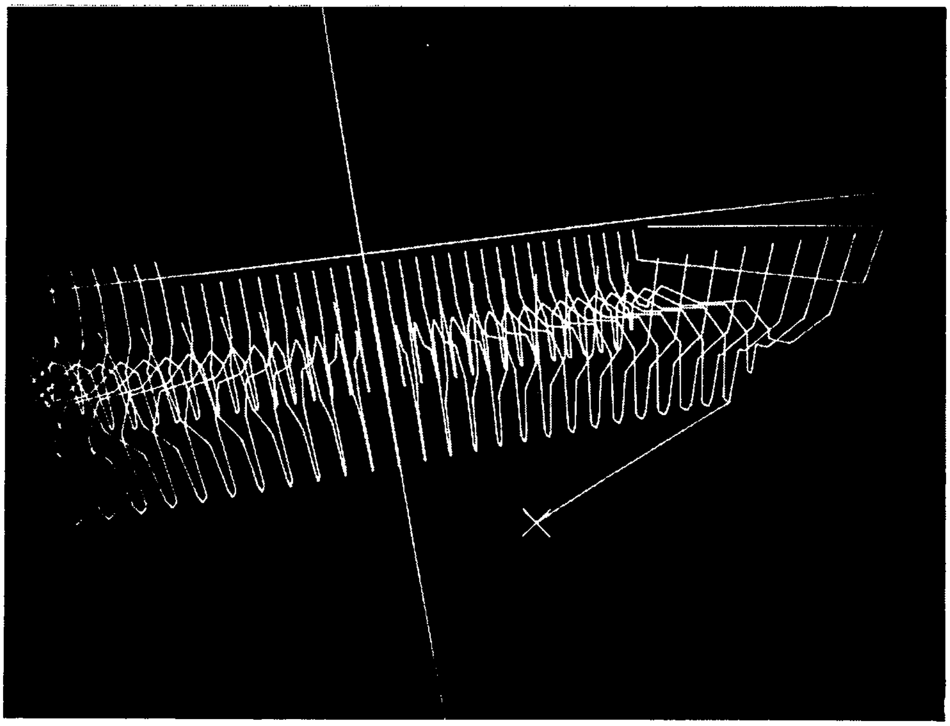


Figure 6. Rubber banding longitudinal lines.

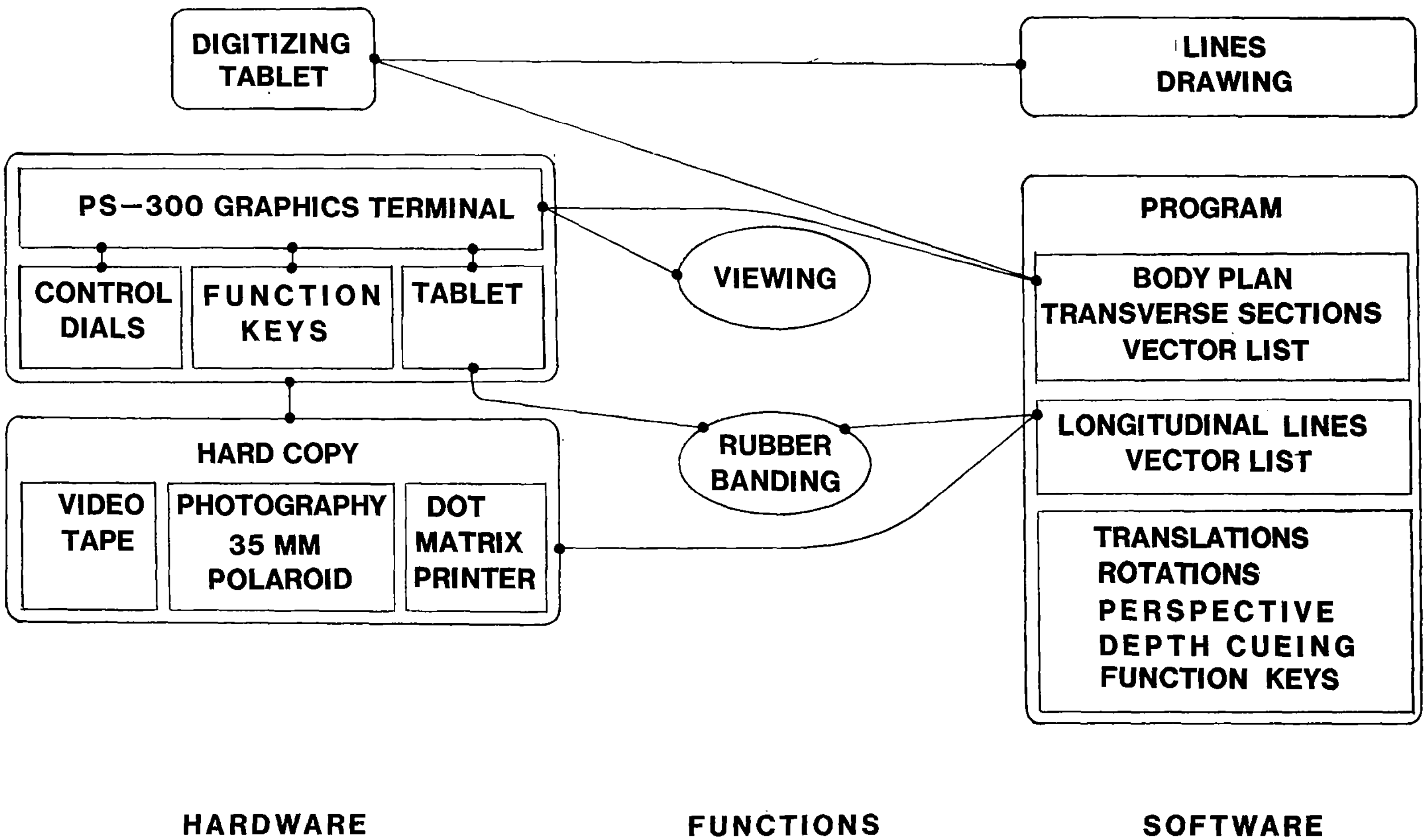


Figure 7. Hullform generation procedure.

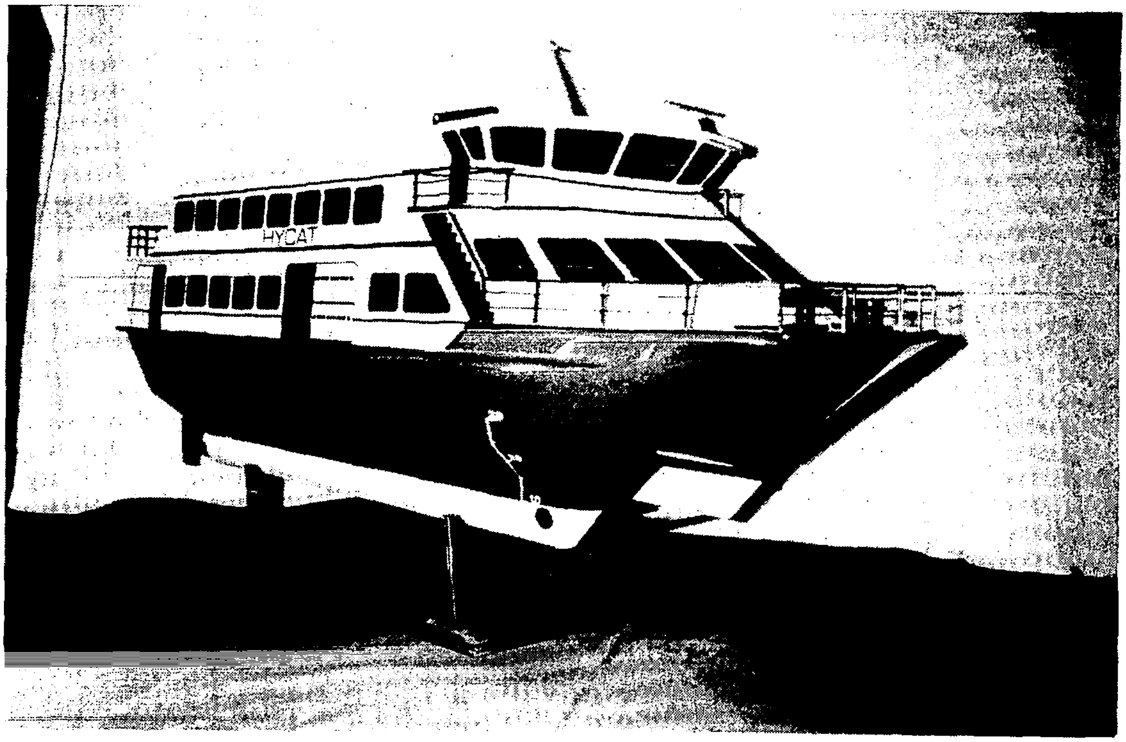


Figure 8. HYCAT - HYdrofoil CATamaran.

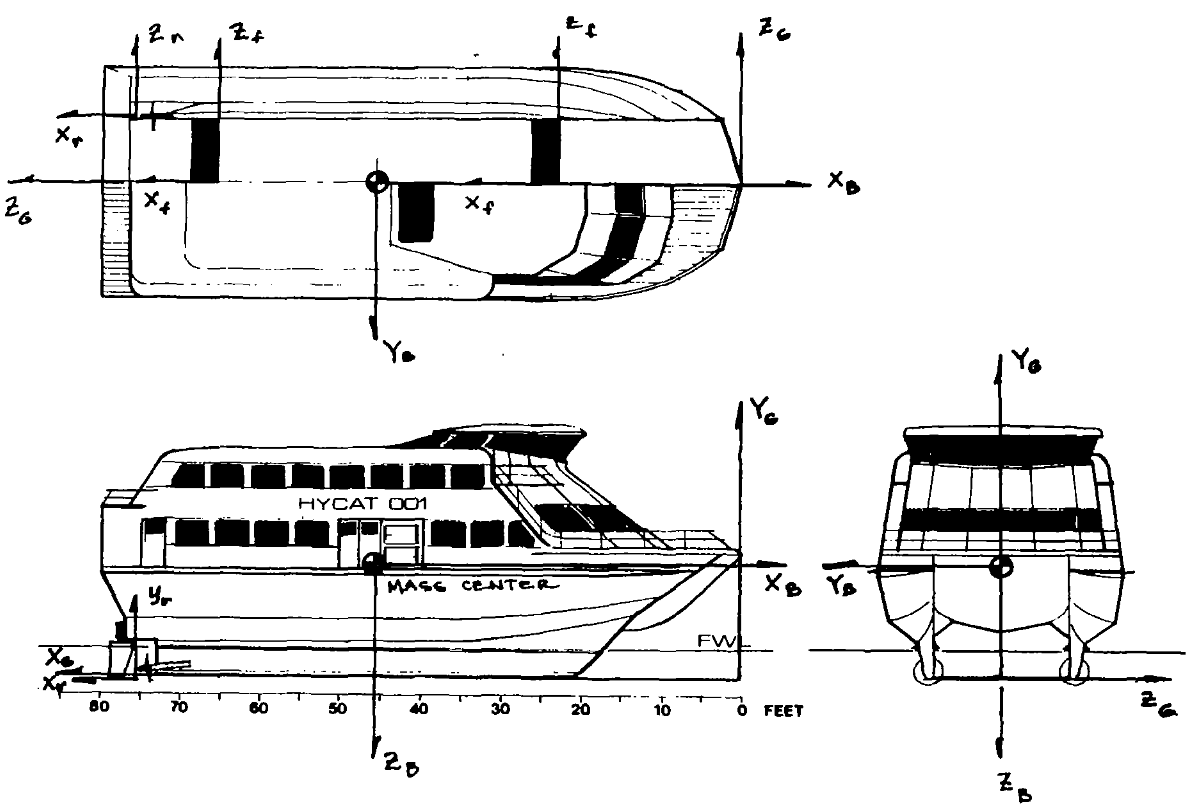


Figure 9. HYCAT - three view.

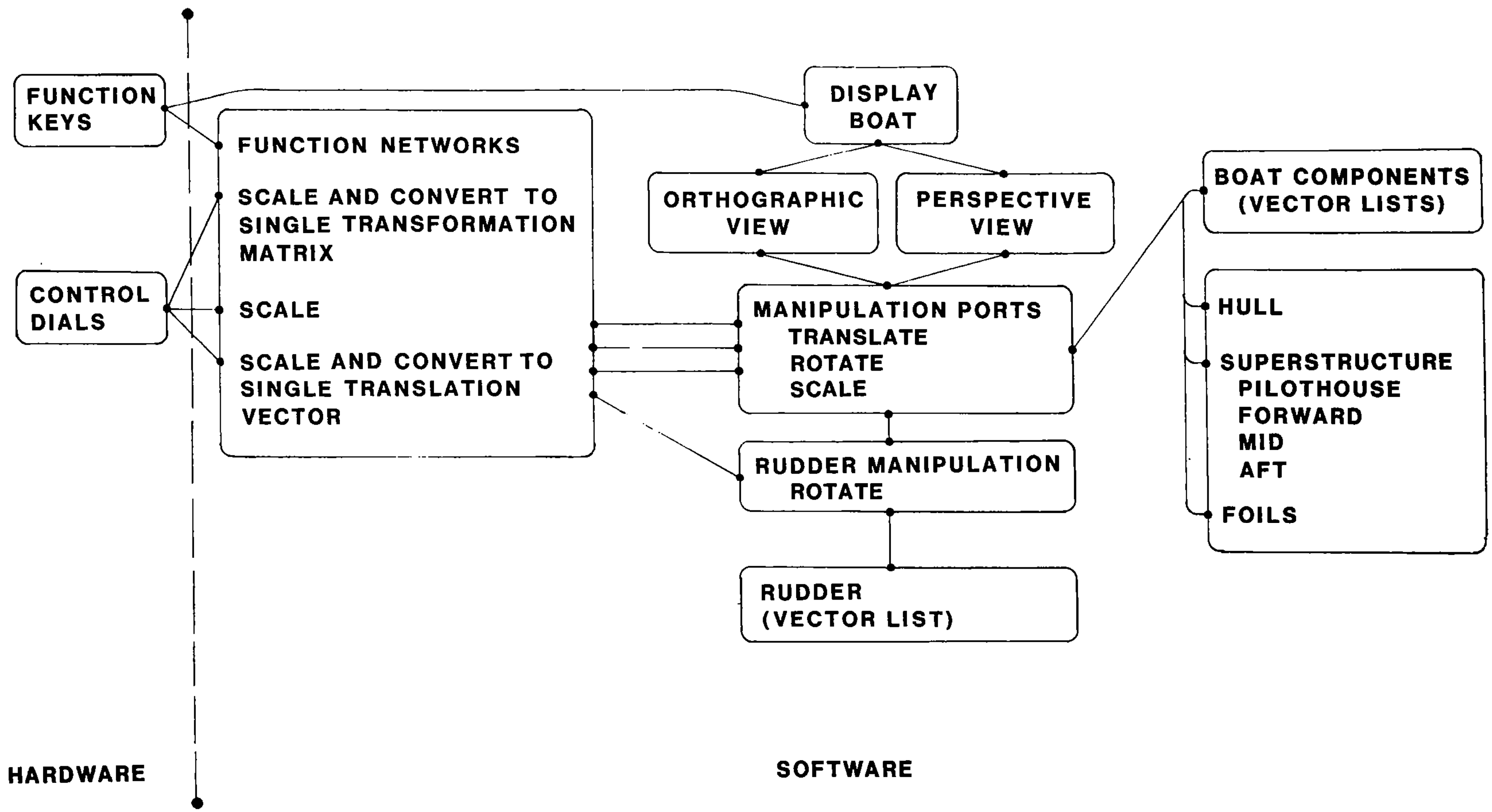


Figure 10. Function networks and data structures.

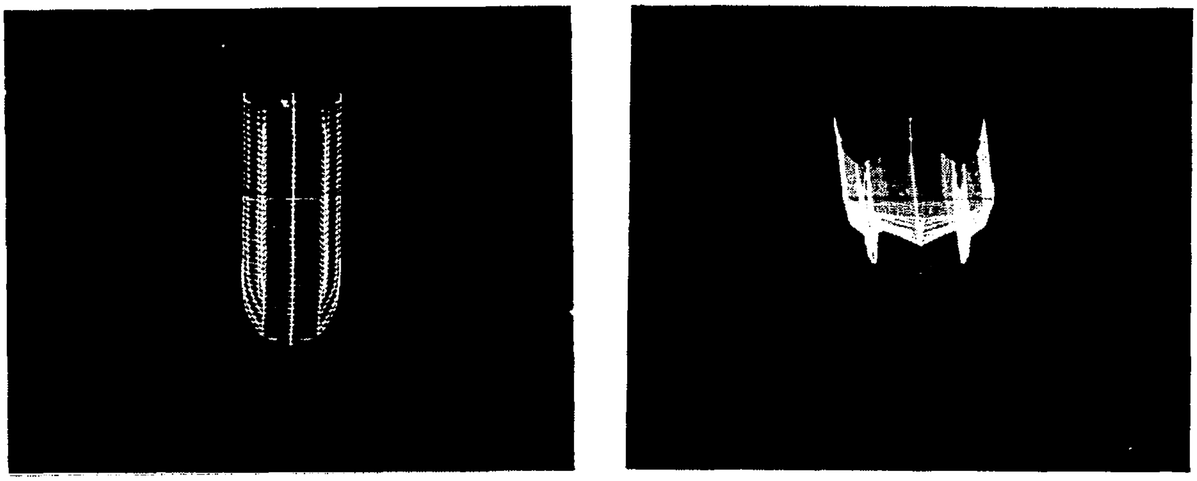


Figure 11. HYCAT axonometric projection (polaroid).

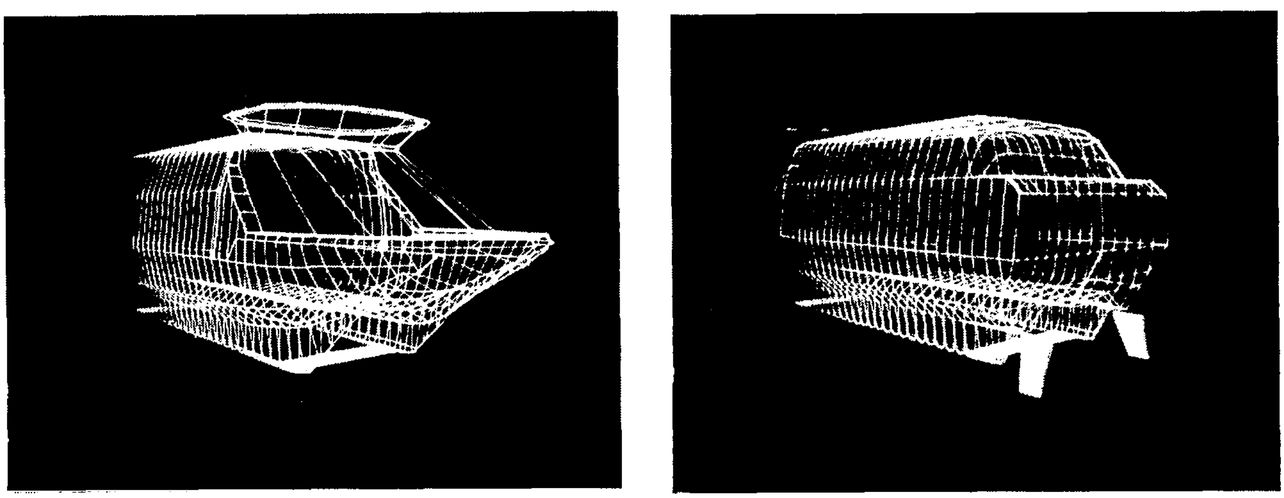


Figure 12. HYCAT perspective projection (35 mm).

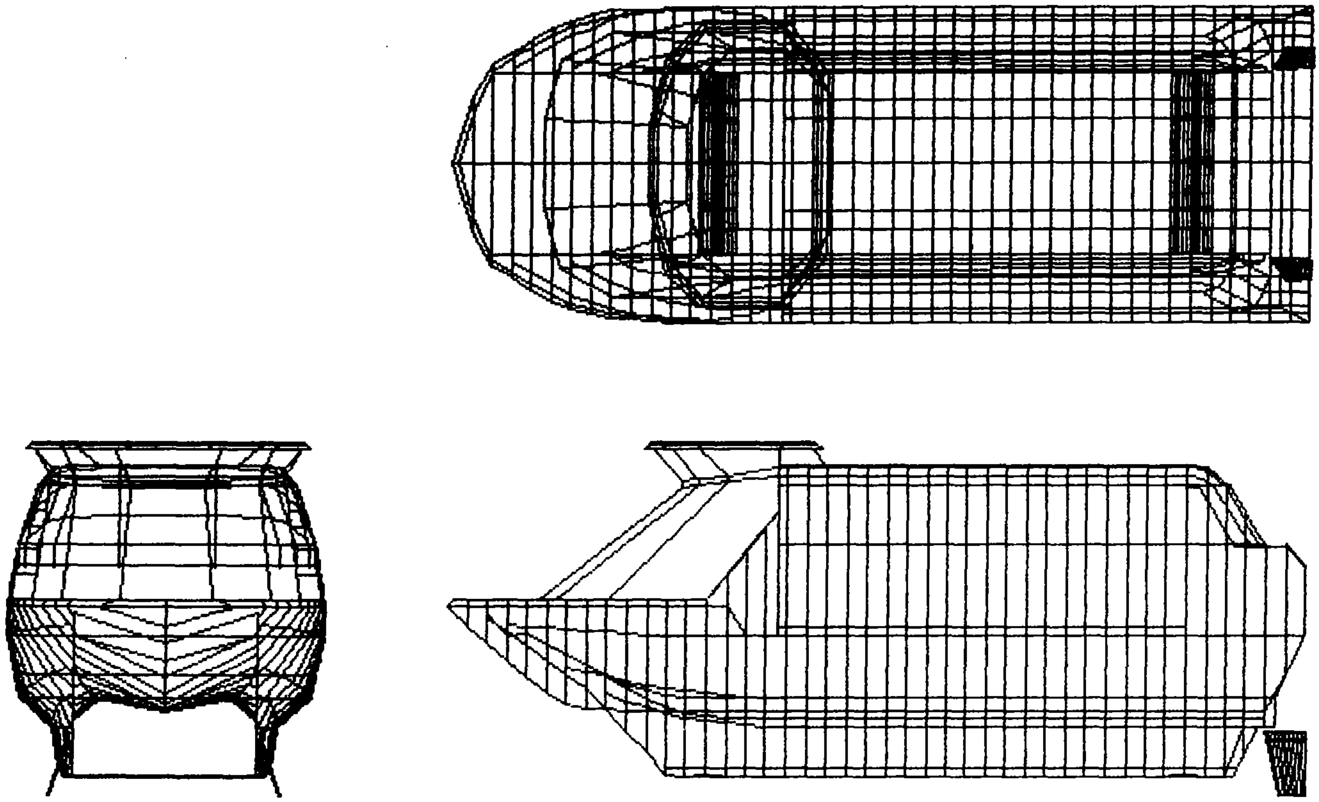


Figure 13. Dot matrix printer - three view.

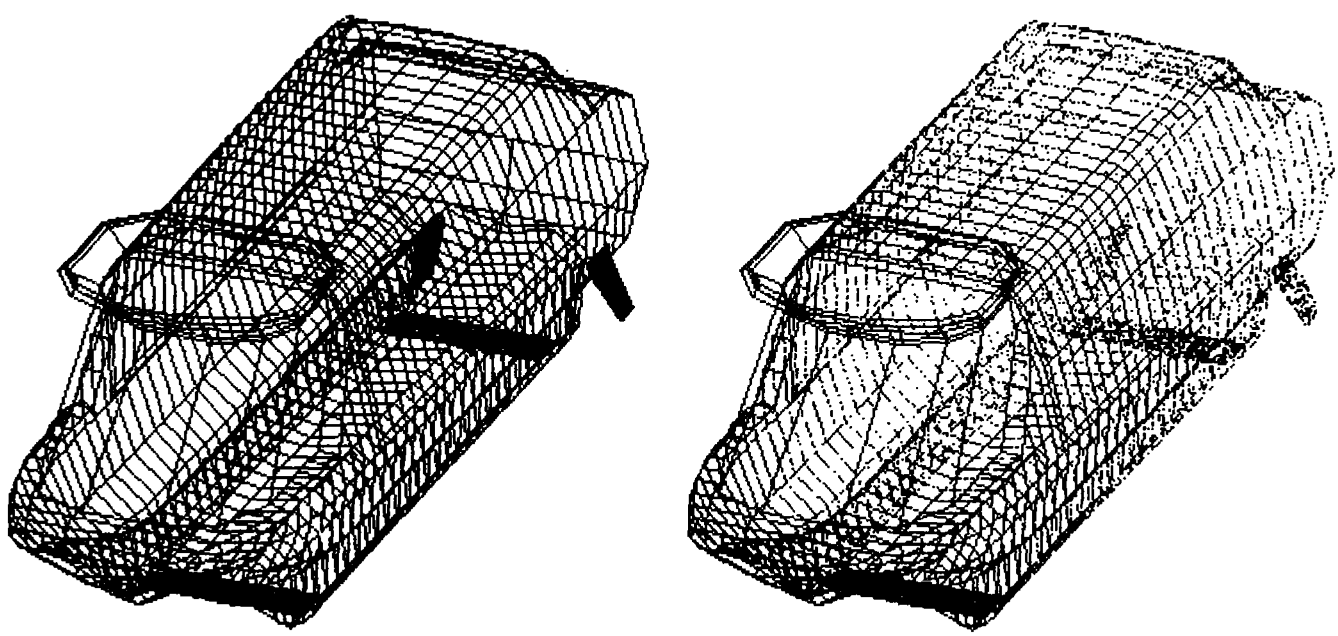


Figure 14. Dot matrix printer - axonometric.

AN INTERACTIVE MODELING PROGRAM FOR THE

GENERATION OF PLANAR POLYGONS FOR BOUNDARY TYPE SOLIDS REPRESENTATIONS

OF WIRE FRAME MODELS

Tulga Ozsoy and John B. Ochs
Department of Mechanical Engineering and Mechanics
Lehigh University
Bethlehem, Pennsylvania

Current three-dimensional CAD/CAM systems that employ wire frame or edge representations of geometry fail to provide surface information necessary for calculating part interferences and component mass properties as well as information for implementing hidden-line or color-shading algorithms. The purpose of the study presented herein has been to develop a general link between 3-dimensional wire frame models and rigid solid models. An interactive computer graphics program has been developed to serve as a front end to a recently announced NASA-developed algorithm (COSMIC Program No. ARC-11446) which offers a general solution to the hidden-line problem where the input data is either line segments or n-sided planar polygons of the most general type with internal boundaries. The developed program provides a general interface to CAD/CAM data bases and has been implemented for models created on the McDonnell Douglas Automation Co.'s Unigraphics VAX 11/780-based CAD/CAM systems with the display software written for DEC's VS11 color graphics devices. The above systems are part of the facilities available in Lehigh University's CAD/CAM laboratories.

Three-D wire frame modeling techniques have been used for at least 10 years by most of the commercially available CAD systems. Geometric modeling by these wire frame systems is viewed by many users as highly interactive and well supported with dimensioning, finite-element mesh generation and machining packages. On the other hand, when the model is constructed as a wire frame, there exist increasing visualization problems of actual models with the increasing complexity of the geometry. Constructing a "hidden-line" representation usually includes surface definitions (as in figure 1), meshing a part with elements, or actually corrupting the model by editing entities. In each case these activities are time consuming, and in addition, features offered by solid modeling systems like sectioning, volume, and other mass property calculations cannot be performed directly on wire frame models. The aim of this study has been to remove these deficiencies by the development of a system to convert wire frame models to a set of planar polygons for boundary type solids representations. With the existence of such a system, wire frame data bases can easily be linked to many solid modeling systems. This is especially important for users who have existing 3-D wire frame data bases. The interactive modeling program is a first step to provide a general conversion system.

The interactive modeling program has been developed on a VAX 11/780 computer in the CAD Laboratory of the Department of Mechanical Engineering and Mechanics at Lehigh University. The program has been implemented on DEC's VS11 raster terminals with 512 × 480 addressable points and 16 colors and is supported by a Lehigh-developed FORTRAN-callable graphics package for VS11 terminals. The VS11-3D Graphics Package allows users full input device control, 3-D dynamic transformations, creation and editing of points, lines, arcs, conics, and splines, and various types of surface representations. The wire frame models are constructed on McAuto's Unigraphics CAD/CAM system and then transferred into a FORTRAN file using a Lehigh-developed transfer program which in turn is used as the input file to the interactive modeler.

Under the developed program, the user can edit the existing wire frame model and when necessary can create, list, purge, and merge the resulting polygons. During the execution of the program, color effects, dynamic transformations, and menu-driven user interactions are provided to make the system user friendly and effective. The input file to the hidden-line program is created automatically. During the transfer of data, the curves are segmented and surface-type polygons are "tiled" to meet the planar-type polygon representation required by the hidden-line code.

Wire frame models are built from lines, arcs, and splines. Each entity end point is defined and merged within specified tolerances to meet the continuous boundary conditions for polygons. Thus, after the 3-D data base is validated, planar polygons can be automatically defined by searching for the common end points of lines (straight or curved) and checking whether they lie in the defined plane or not. If the searching algorithm is not able to define a closed polygon starting from one line, additional searches are applied in different directions at common nodes. Each newly defined polygon is checked against previous ones to prevent redefinition. Polygon edges change color after they have been defined. Any line shared by two different polygons is indicated by a certain color and is classified as a model boundary.

Redundancies and errors in the original 3-D wire frame model are flagged by color for the user to edit interactively. This editing capability is part of the interactive modeler and allows the user to construct a consistent model. Edges of the geometry that are nonplanar are flagged and require the user to select either automatic or manual surface definitions (fig. 2). The available surface types include ruled surfaces, tabulated cylinders, surfaces of revolution and sculptured surfaces. Surface-type polygons are then automatically generated from the selection of the surface boundary curves.

With the completion of the generation of the planar polygons for the entire model, the user can invoke the NASA hidden-line algorithm. The resulting hidden-line representation can be displayed in different colors and surface tiles can be switched on or off (fig. 3). The interactive modeler allows any edge representation to be viewed from any point and to be scaled or zoomed during the execution of the program. Interactive menus linked in a tree-like structure allow users a natural way of communication with the system. For internal boundary representations, a polygon merge option has been added. With this option, holes can be defined by the selection of the polygons to be merged (fig. 4). Polygons can be listed to investigate and validate their representations, or can be purged interactively. In order to generate the surface normals in the proper sense for boundary representations schemes such as GEOMOD[1], the user can integrate the ordering of end points in any polygon and if necessary automatically reorder them in a clockwise or counterclockwise fashion, as shown in figure 5. With the polygon merging and ordering complete, a file can be written which will format the data for input to GEOMOD or MOVIE.BYU. Figure 6 shows the shaded representation of the model with an additional hole generated by Boolean operations in GEOMOD.

---

[1]System Design and GEOMOD are products of Structural Dynamics Research Corporation of Milford, Ohio.

A general IGES (Initial Graphics Exchange Specification) interface is currently under development. Future enhancements will include providing a post-processor within the NASA hidden-line code to eliminate back faces, and the development of the logic to eliminate the redrawing of edges twice to provide dashed-line representation of hidden features. Additionally, an interface program is being developed to return the two-dimensional hidden-line representation to the wire frame CAD system for documentation and dimensioning.

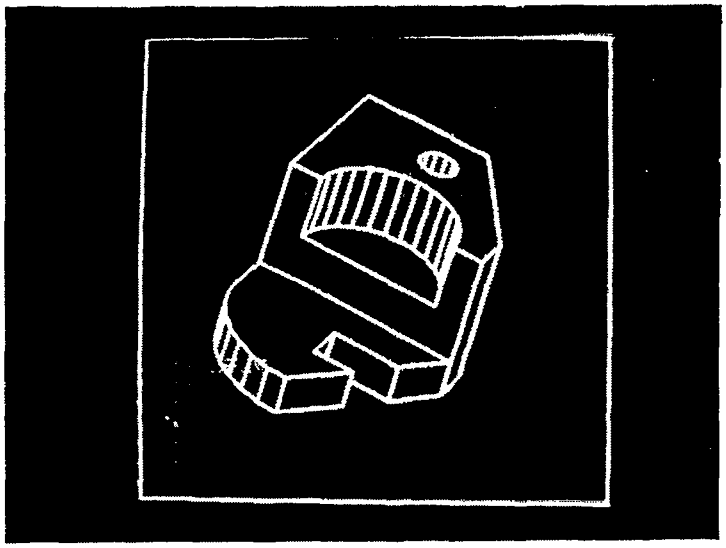


Figure 1.- Hidden lines have been removed and "surface tile" option is active.

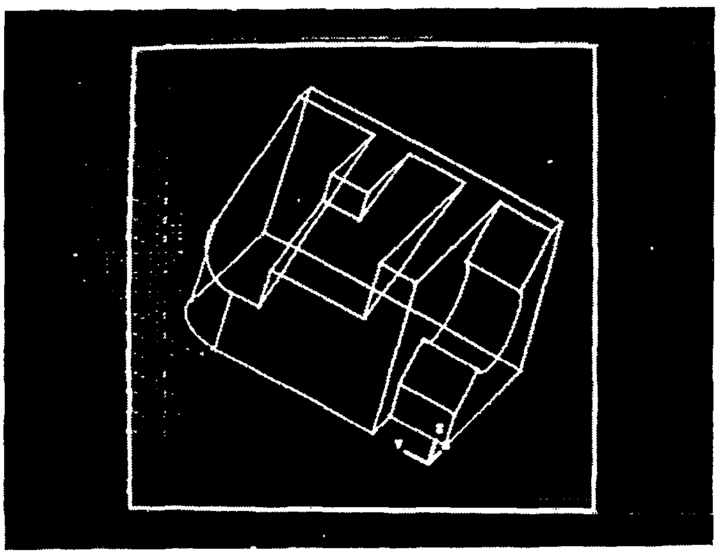


Figure 2.- All planar polygons have been defined. Two surface-type polygons (curved lines) are flagged to be defined by the user.

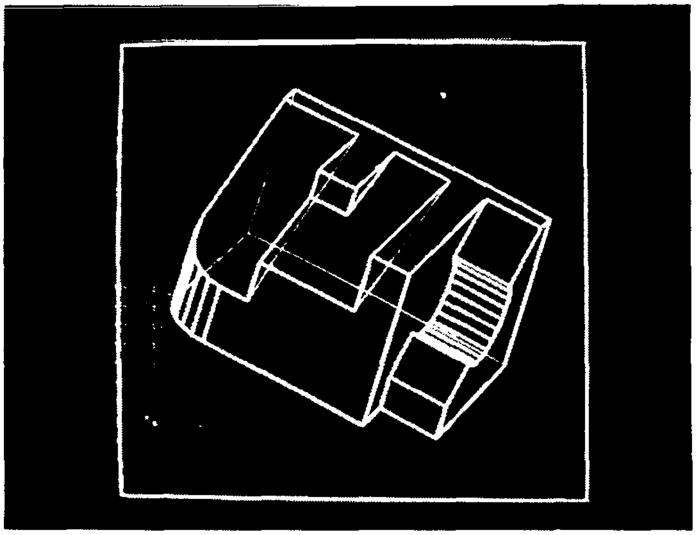


Figure 3.- Hidden-line code has been applied. Surface tiles and hidden lines with different color options are displayed.

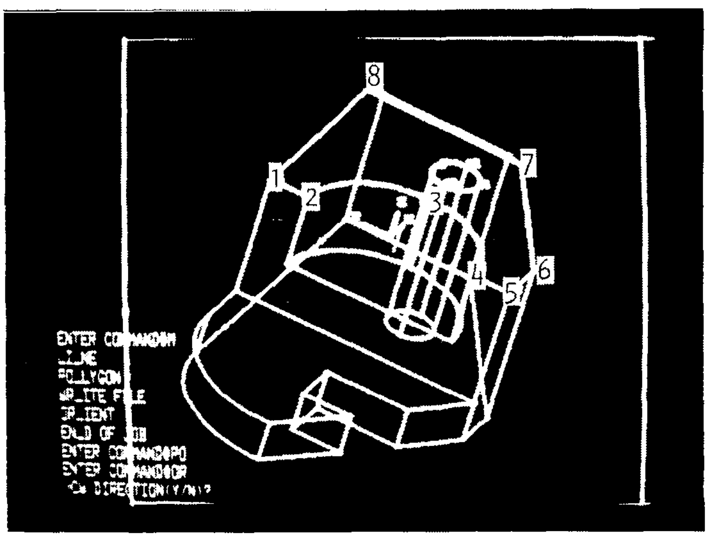


Figure 4.- Polygons are merging to allow internal boundary definition for hole representation, as numbered in figure.

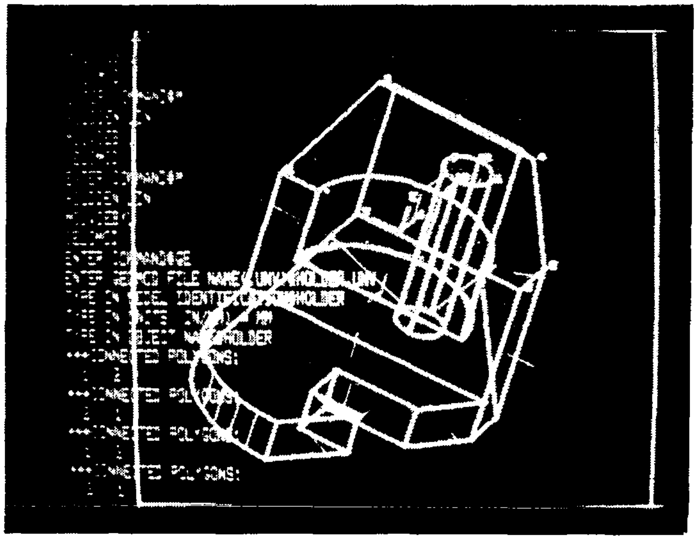


Figure 5.- Polygons have been merged and ordered for the proper surface-normal definitions for GEOMOD and MOVIE.BYU.

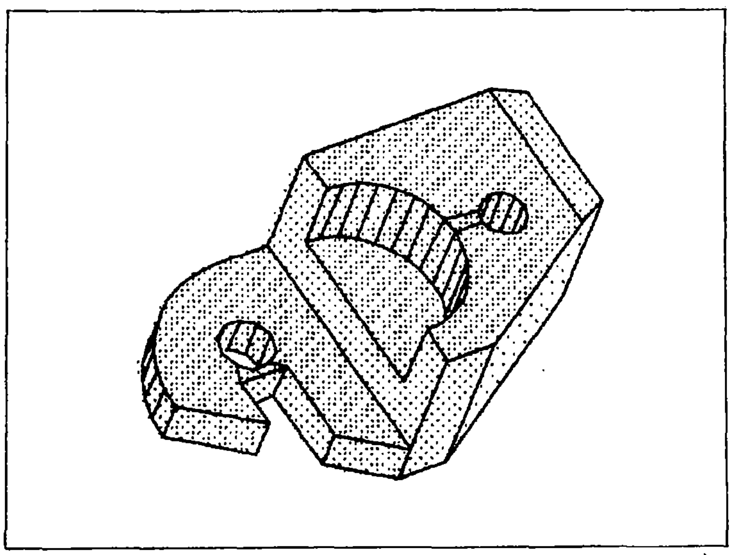


Figure 6.- Shaded representation of the model with additional holes generated by Boolean operators within GEOMOD.

MANAGING GEOMETRIC INFORMATION

WITH A DATA BASE MANAGEMENT SYSTEM

R. P. Dube

Boeing Computer Services, Seattle, WA

This report describes the strategies for managing computer-based geometry. The computer model of geometry is the basis for communication, manipulation, and analysis of shape information. The research on IPAD (Integrated Programs for Aerospace-Vehicle Design) has focused on the use of data base management system (DBMS) technology to manage engineering/manufacturing data. The data base system then makes this information available to all authorized departments.

One of the objectives of IPAD is to develop a computer-based engineering complex which automates the storage, management, protection, and retrieval of engineering data. In particular, this facility must manage geometry information as well as associated data. In this report, we will discuss the approach taken on the IPAD project to achieve this objective. We will begin by examining geometry management in current systems and the approach taken in the early IPAD prototypes. We will then shift our attention to the ongoing IPAD work.

Current geometry systems (i.e., CAD turnkey systems) manage all the geometry information for one individual internally to the system. Satisfying requirements for geometry information outside the system for many users can be very difficult. Two commonly used methods to overcome this problem involve either adding new application programs to the geometry system or using pre- and post-processing programs to access the geometry. Neither of these methods is very promising for a long-term solution.

Some of the early work done on IPAD investigated the combination of a relational data manager and software drivers specific to geometry. This work involved RIM, the relational DBMS produced by IPAD, and a library of Pascal subroutines. Essentially, RIM is used to manage the geometry data (records) while the software drivers are used to manage the geometry information. This strategy has some nice features. The data manager and geometry drivers may be linked with the applications requiring geometry information. Its major drawback, however, is that any changes in the geometry records or model require an actual change in the software drivers. A more promising solution has been developed in the current IPAD work. This work has eliminated the need for the software drivers, thus allowing for the definition and manipulation of the geometry information through the DBMS languages.

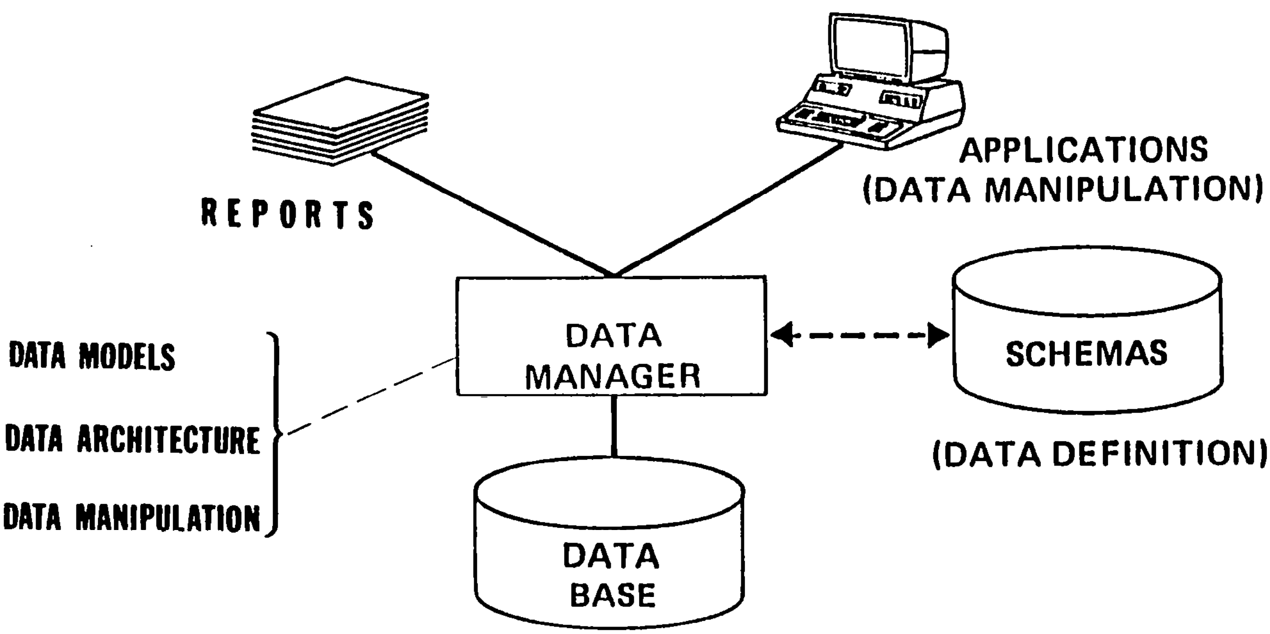
The cornerstone of the IPAD concept is its information processor (IPIP). IPIP is a data base management system that manages the data required by and produced by groups of people during the design process. Since IPIP focuses on the need for information to be shared, its capabilities are oriented to managing the integration of the engineering data. IPIP capabilities range from support for multiple schemas and data models to support for data inventory management. IPIP stresses integration of functionality in terms of uniformity of user languages, commonality of capabilities, and an integrated software architecture. User languages reflect integration through a unified approach to multiple data models and multiple application environments in a single logical schema language, a single internal schema language, a single mapping schema language, a single data manipulation language; and through commonality between the logical and internal schema languages. Through the use of IPIP's data definition languages, the structure of the geometry components is defined. The data manipulation language is the vehicle by which a user defines an instance of the geometry. The manipulation language also allows a user to edit, query, and manage the geometry.

DATA BASE MANAGEMENT SYSTEM (DBMS)

- o ASSUMPTION - DBMS TECHNOLOGY IS A POWERFUL TOOL WHICH WILL ALLOW THE INTEGRATION OF THE ENGINEERING AND MANUFACTURING PROCESSES
- o PROBLEM - CURRENT DBMS TECHNOLOGY DOES NOT PROVIDE SUFFICIENT FUNCTIONALITY TO MEET THE REQUIREMENTS; I.E., GEOMETRY, VECTORS, MATRICES, ETC.
- o SOLUTION - DESIGN AND DEVELOP A DBMS WHICH ADDRESSES THE INFORMATION MANAGEMENT PROBLEM OF THE CAD/CAM ENVIRONMENT
- o RESULT - CURRENT AND FUTURE DBMS TECHNOLOGY PRODUCTS WOULD BE AVAILABLE TO ASSIST IN THE INTEGRATION OF THE ENGINEERING AND MANUFACTURING PROCESSES

DBMS DEFINITION

- o DATA BASE MANAGEMENT SYSTEM (DBMS): A UNIFIED DESCRIPTION OF DATA TOGETHER WITH THE MEANS TO MANIPULATE THE DATA



- Separation of data definition and data manipulation

OPERATION OF DBMS

- o A DATA BASE ADMINISTRATOR (DBA) DESCRIBES THE USER DATA WHICH THE DBMS MUST CONTROL
    - o LANGUAGE (ANALOGOUS TO DECLARING VARIABLES IN PROGRAMMING)
    - o DEFINES ITEMS OF DATA AND RELATIONS
- o DESCRIPTION IS WRITTEN USING A DATA DESCRIPTION LANGUAGE (DDL)
    - o RECORD TYPE CORRESPONDS TO AN ENTITY CLASS
    - o RECORD OCCURRENCE DESCRIBES A SPECIFIC ENTITY
    - o ASSOCIATIONS OR RELATIONSHIPS
- o DATA MANIPULATION
    - o APPLICATION PROGRAM OR QUERY USER ACCESSES
    - o OPERATIONS TO ADD, RETRIEVE, REPLACE, OR DELETE RECORDS MAY BE PERFORMED ONLY ON DATA DESCRIBED BY THE DBA
- o MANAGEMENT SERVICES
    - o BACKUP AND RECOVERY
    - o SECURITY
    - o DATA SET MANAGEMENT
    - o CONFIGURATION CONTROL

IPAD* R & D

STEP 1: EVALUATE THE DATA STRUCTURES AND FLEXIBILITY OF THE "TURNKEY" GEOMETRY SYSTEMS (1978)

STEP 2: EVOLVE AN INITIAL DBMS SOLUTION COMPOSED OF SOFTWARE DRIVERS AND A CONVENTIONAL DBMS (1979)

STEP 3: DEVELOP A SPECIFICATION FOR GEOMETRY MANAGEMENT INTEGRATED AS PART OF THE DBMS (1980)

STEP 4: DESIGN AND DEVELOP THE IPAD INFORMATION PROCESSOR (IPIP)

*INTEGRATED PROGRAMS FOR AEROSPACE-VEHICLE DESIGN (IPAD)
NASA CONTRACT NO. NAS1-14700 TO THE BOEING CO.

STEP 1 - EVALUATE CURRENT GEOMETRY SYSTEMS

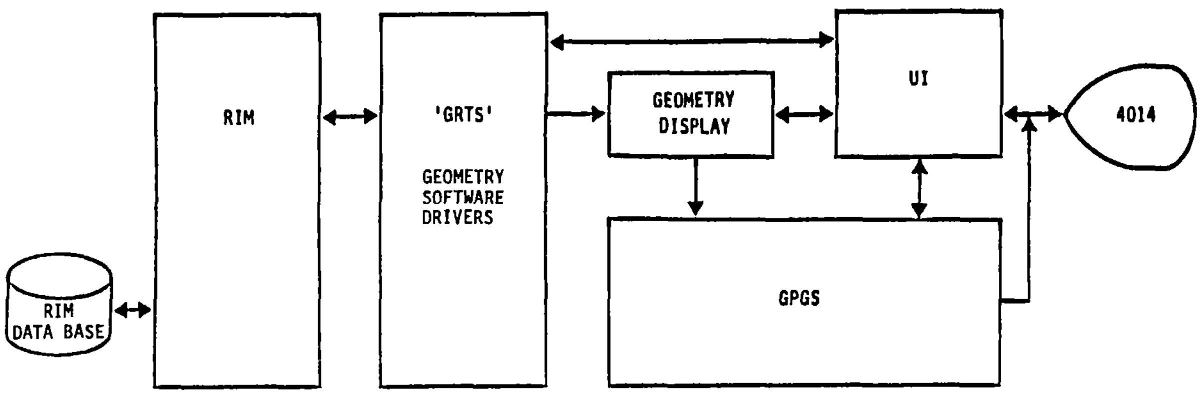
- o GEOMETRY INFORMATION MANAGED INTERNALLY AND AVAILABLE ONLY TO USERS OF THE SYSTEM
- o MOST GEOMETRY DATA IS STORED IN TABLES
- o OPTIONS FOR AVAILABILITY OF GEOMETRY TO OTHER APPLICATIONS
  - o PRE- AND POST-PROCESSORS
  - o EMBED THE APPLICATION IN THE EXISTING SYSTEM
- o IGES (INITIAL GRAPHICS EXCHANGE SPECIFICATION) USED AS A GEOMETRY COMMUNICATION STANDARD

STEP 2 - INITIAL DBMS SOLUTION

- o RIM (RELATIONAL INFORMATION MANAGER*) USED TO MANAGE THE NUMERICAL DATA
- o SOFTWARE DRIVERS DEVELOPED TO MANAGE THE GEOMETRY INFORMATION
- o RESULT
  - o GEOMETRY IS "CODED IN"
  - o SOFTWARE DRIVERS LOGICALLY BECOME PART OF THE DBMS
- o ADVANTAGE - GEOMETRY IS DIRECTLY ACCESSIBLE FROM THE DATA MANAGER

*DEVELOPED BY IPAD (1978); CURRENTLY A COMMERCIAL PRODUCT

INITIAL PROTOTYPE - 1979

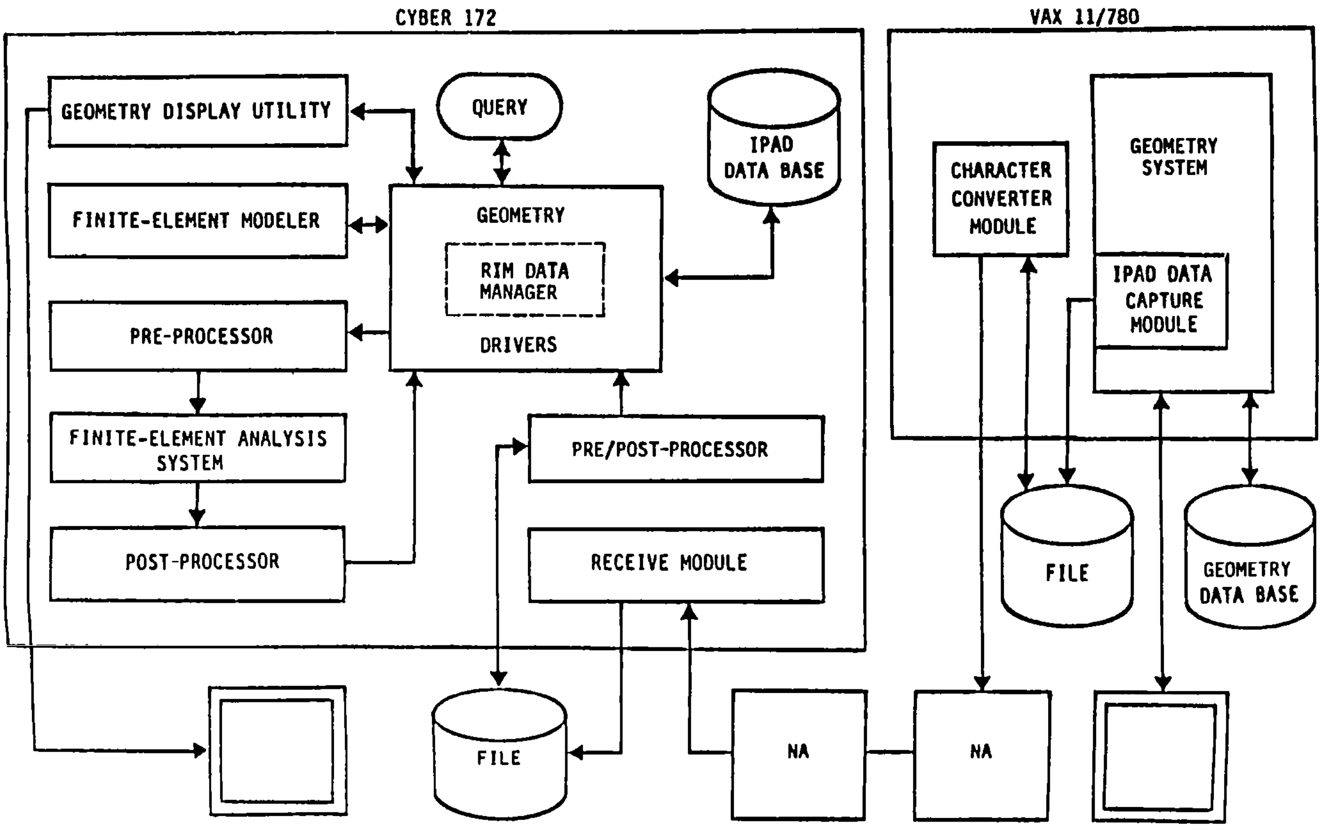


RIM	-	RELATIONAL INFORMATION MANAGER (FORTRAN)
GPGS	-	GRAPHICS SUBSYSTEM (FORTRAN)
UI	-	PROTOTYPE USER INTERFACE (PASCAL)
GRTS	-	PROTOTYPE GEOMETRY RUNTIME SYSTEM (PASCAL)

GEOMETRY RUNTIME SYSTEM - CAPABILITIES

DEFINE	PRIMITIVE ENTITIES, TRANSFORMATIONS
MODIFY	VALUES DEFINING A PRIMITIVE ENTITY
DELETE	AN ENTITY OR OBJECT
COPY	AN ENTITY OR OBJECT
RENAME	AN ENTITY OR OBJECT
INITIATE	A COMPOSITE ENTITY
JOIN	A SEGMENT TO A COMPOSITE CURVE, OR A PATCH TO A SURFACE
EXTEND	A COMPOSITE ENTITY
REMOVE	AN ENTITY FROM A COMPOSITE ENTITY OR FROM AN OBJECT
BUILD	AN OBJECT FROM ENTITIES OR OTHER OBJECTS
LIST	COMPONENTS ON AN ENTITY OR OBJECT
GET	AN ENTITY OR OBJECT

IPAD INTEGRATION PROTOTYPE - 1979



STEP 3 - DEFINITIONS

- o GEOMETRY ENTITIES: IDENTIFIABLE PIECES OF GEOMETRY; POINTS, LINES, CONIC SEGMENT, ETC.
- o SCHEMA: THE ORGANIZATION OF THE DATA BASE.
- o SETS: IDENTIFIES WHICH RECORDS ARE ASSOCIATED IN AN OWNER/MEMBER RELATIONSHIP.
- o STRUCTURE: IPAD UNIQUE PROCESSING TO AGGREGATE DISCRETE PIECES OF DATA TO A GEOMETRY ENTITY.
- o DDL: DATA DEFINITION LANGUAGE DESCRIBES THE ORGANIZATION OF THE DATA BASE.
- o DML: DATA MANIPULATION LANGUAGE DESCRIBES THE WAY THE DATA BASE IS PROCESSED.

GEOMETRY SPECIFICATIONS IN IPIP DESIGN--SUMMARY

- o GEOMETRIC IDEAS EXPLICIT IN DDL, DML, AND SYSTEM PROCESSING
  - o PRIMITIVES - VECTORS (POINTS, TANGENTS, ETC.) AND VERTICES
  - o AGGREGATES OF PRIMITIVES - FORMING ENTITIES - LINES, ARCS, SURFACES
  - o AGGREGATES OF ENTITIES - FORMING OBJECTS - PARTS, ETC.
- o MATH OF GEOMETRY INDEPENDENT OF IPIP (DBMS)
  - o DEFINABLE BY USER
  - o MANIPULATION (OF IPIP RETRIEVED DATA) BY USER APPLICATION
- o GEOMETRY DATA HAS ALL BENEFITS OF OTHER DATA IN IPIP
  - o SHARED (NO REDUNDANT DATA)
  - o MANY IPIP ACTIONS PER SINGLE (HIGH LEVEL - I.E., PART) REQUEST
  - o INTEGRITY OF DATA ITSELF AS WELL AS SCHEMAS, ETC.

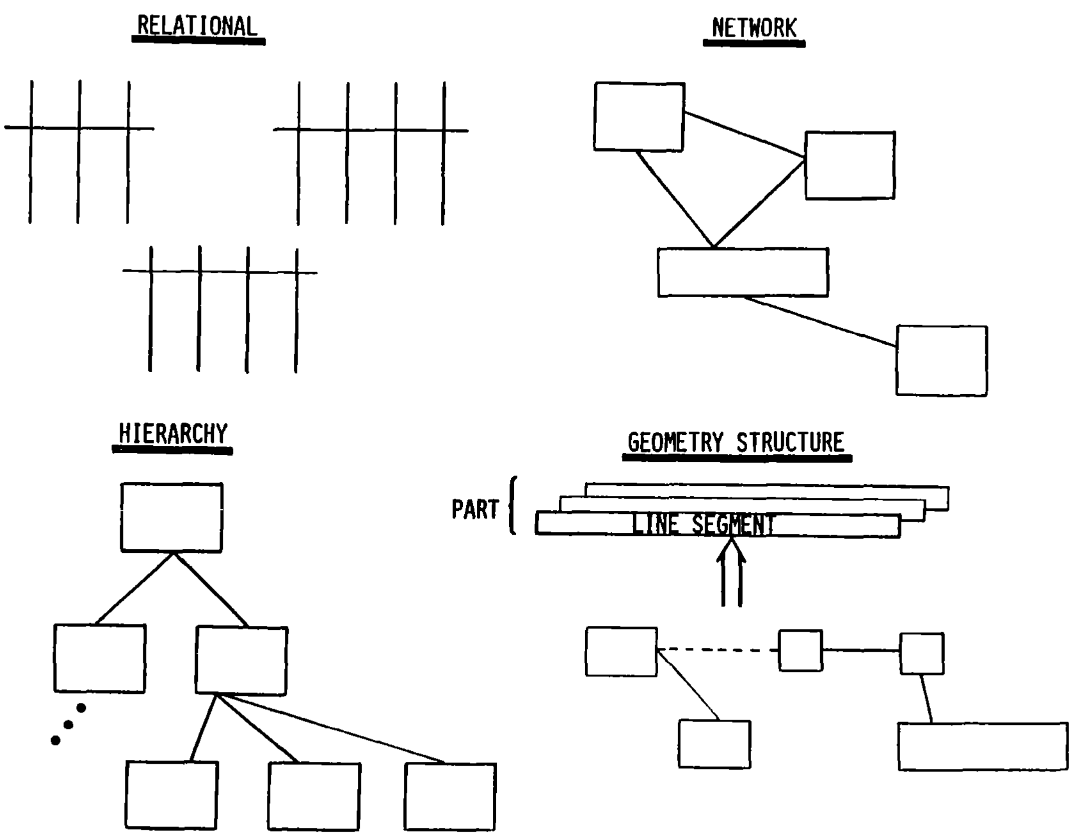
STEP 4 - GEOMETRY DATA MANAGEMENT IN IPIP

- o GOAL
  - o SYSTEM LEVEL GEOMETRY MANAGEMENT INDEPENDENT OF APPLICATION PROGRAMS
- o APPROACH
  - o USE EXISTING IPIP FUNCTIONALITY SUPPLEMENTED BY FUNCTIONS REQUIRED TO SUPPORT THE GEOMETRY MODEL
  - o GEOMETRY DATA STRUCTURES DEFINED THROUGH DATA DEFINITION LANGUAGE (DDL)
  - o GEOMETRY, LIKE OTHER DBMS DATA, IS "SOFT CODED" INTO THE SYSTEM
  - o NEW GEOMETRY MAY BE ADDED THROUGH ADDITION OF NEW RECORDS, ASSOCIATIONS, STRUCTURES
  - o FACILITIES ARE GENERAL AND MAY BE USED BY ALL DATA BASE USERS
- o RESULT
  - o MULTI-MODEL INTEGRATED DATA MANAGEMENT SYSTEM

DATA MODEL - IPIP

- o SYSTEM IS DESIGNED TO SUPPORT
  - o RELATIONAL MODEL
  - o NETWORK MODEL
  - o HIERARCHICAL MODEL
  - o GEOMETRY MODEL
- o RELATIONAL AND NETWORK FEATURES INTEGRATED
- o ONE FAMILY OF DATA DEFINITION LANGUAGES
  - o INTERNAL SCHEMA LANGUAGE
  - o LOGICAL SCHEMA LANGUAGE
  - o MAPPING SCHEMA LANGUAGE
- o DATA MODEL ALLOWED AT ANY LEVEL OF THE LOGICAL SCHEMA
- o CONSISTENT WITH DBMS STANDARDS

M O D E L S

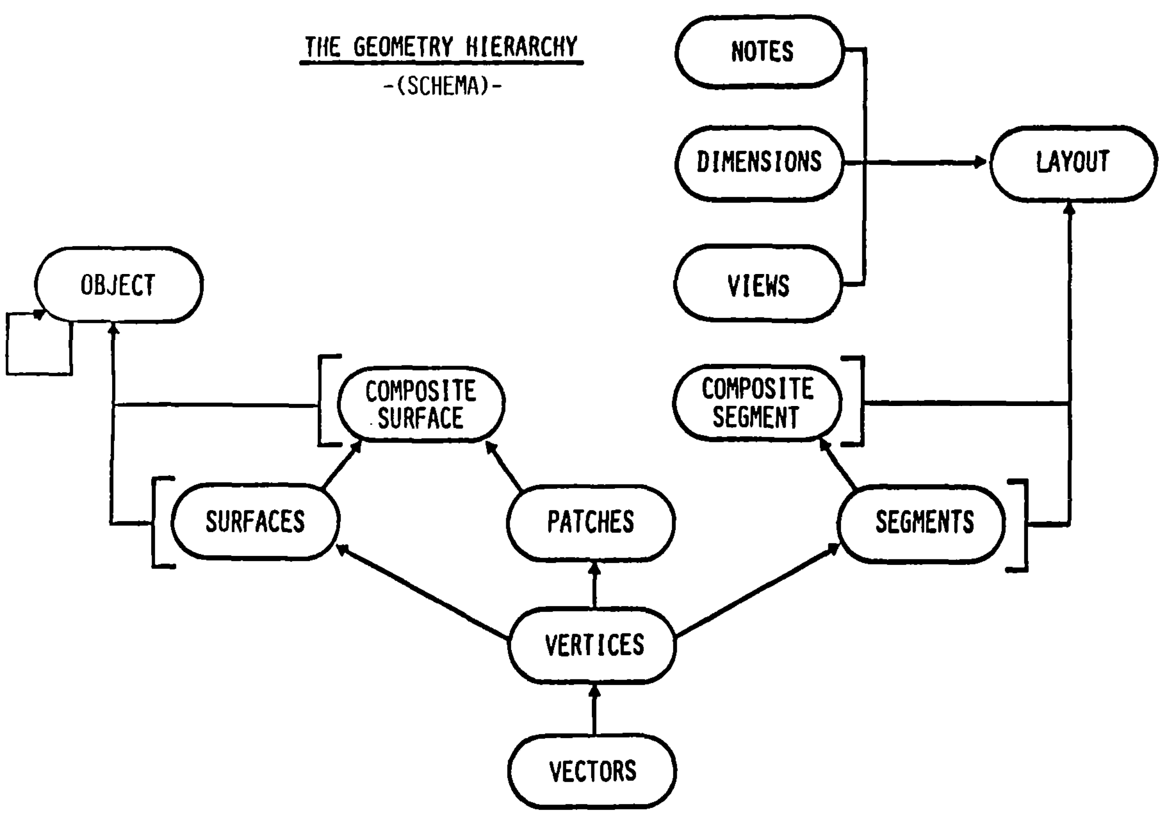


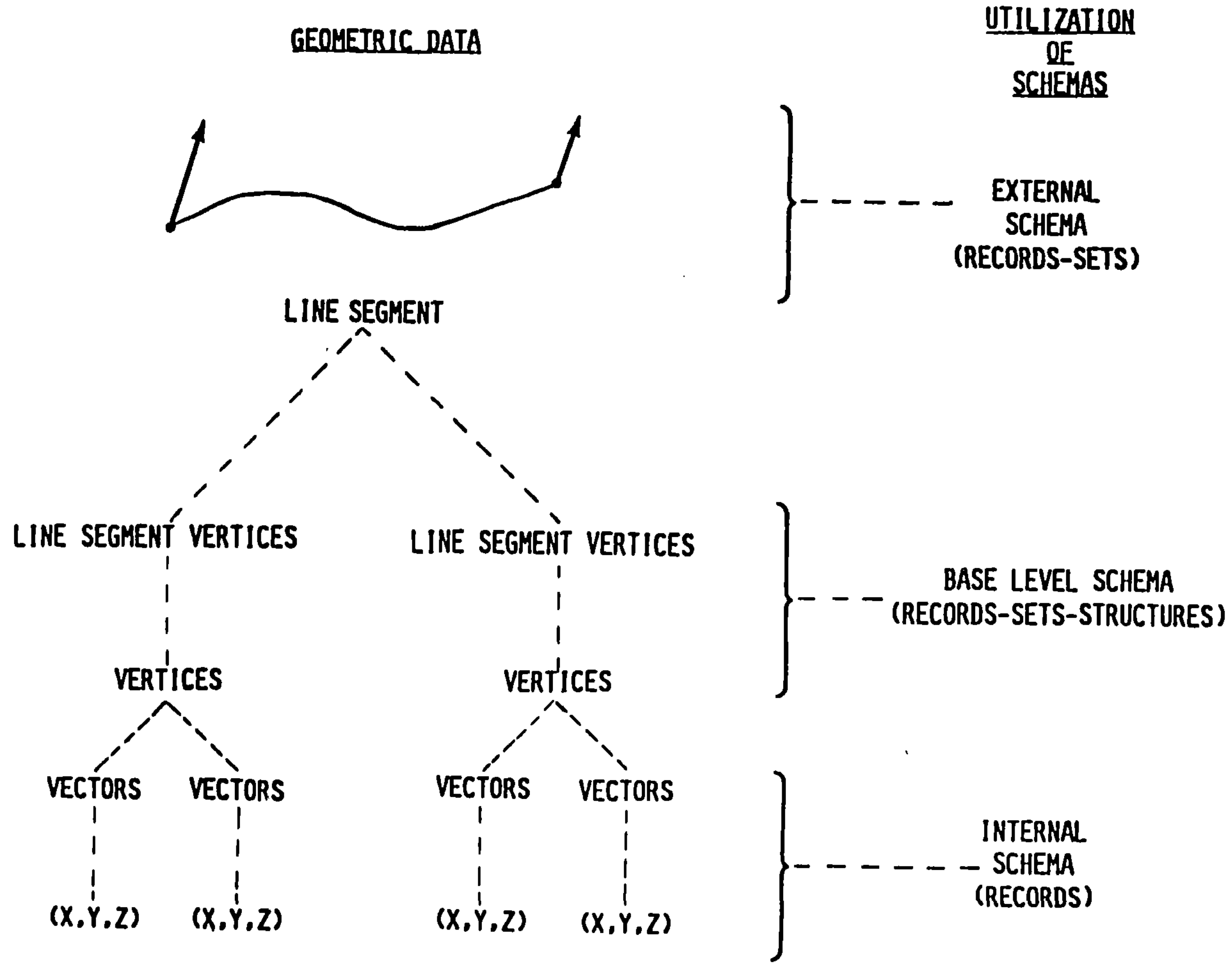
IPAD GEOMETRY DATA MANAGEMENT

- o DEFINING DATA FOR IPAD CANONICAL FORMS DESCRIBED IN BASE-LEVEL SCHEMA
- o ASSOCIATIONS BETWEEN DATA ITEMS ARE DEFINED
- o RECORDS OF BASE-LEVEL SCHEMA ARE GROUPED BY EXPLICIT SCHEMA CONSTRUCT CALLED A STRUCTURE
- o MAPPING SCHEMA PROVIDES CORRESPONDENCES BETWEEN EXTERNAL SCHEMA ITEMS AND ITEMS IN STRUCTURE
- o ACCESS TO ENTITIES ARE MADE BY DML OPERATIONS ON EXTERNAL RECORD
- o DML SEMANTICS ARE CONSISTENT WITH GEOMETRY REQUIREMENTS
- o DML PROCESSING DIRECTIVES ARE LARGELY BASED ON ASSOCIATIONS (SETS)
  - o SOURCE
  - o DELETE MEMBERS IF DELETE OWNERS
  - o COPY MEMBERS IF COPY OWNER
  - o DELETE OWNER IF SET EMPTY

THE GEOMETRY HIERARCHY

-(SCHEMA)-



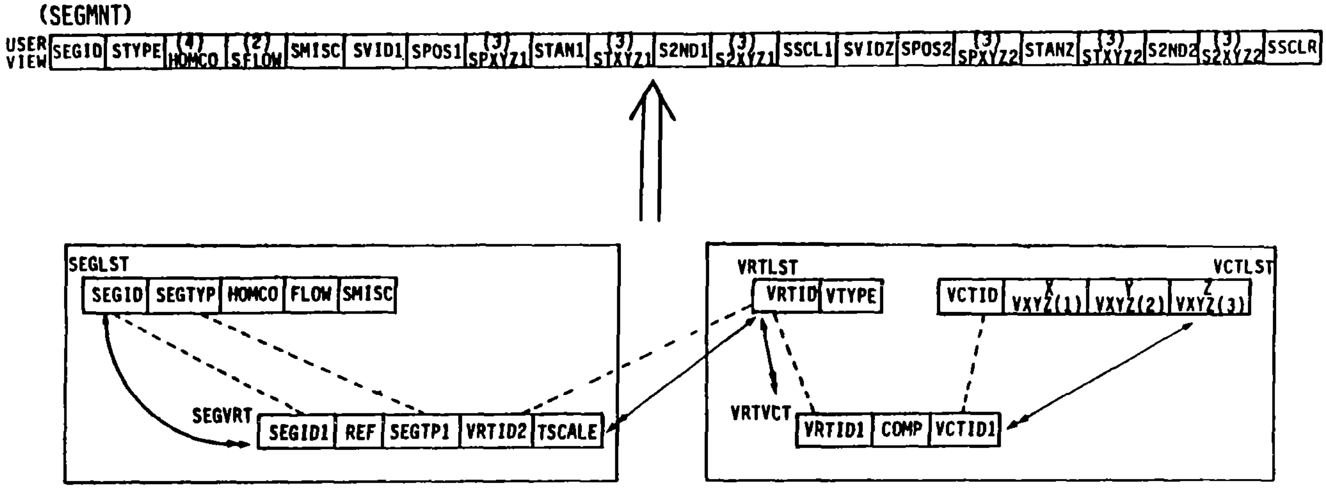


DATA MANIPULATION

- o PUT DML STATEMENTS IN APPLICATION PROGRAM
- o RUN PRECOMPILER
  - o CHECKS SYNTAX AND SEMANTICS OF DML STATEMENTS
  - o STORE INFORMATION FOR IPIP
  - o CHANGES DML STATEMENTS TO COMPILABLE FORTRAN CODE
- o RUN FORTRAN COMPILER, LINK, AND EXECUTE
- o ONE DATA MANIPULATION LANGUAGE AVAILABLE FOR ALL MODELS

- o GEOMETRY DATA MANIPULATION - (ALL LOGICAL SCHEMAS)
  - o STORE, FIND, FETCH, DELETE
    - o PRIMITIVE ENTITIES
    - o GEOMETRIC ENTITIES (AGGREGATES OF PRIMITIVES)
    - o GEOMETRIC "PARTS" (AGGREGATES OF ENTITIES)
  - o ALLOWS USER MANIPULATION - "AS IF" A SINGLE RECORD
    - o INTERNAL TO IPIP A SERIES OF OPERATIONS OCCURS AGAINST GROUPS OF RECORDS UNDERLYING THE DEFINED STRUCTURE
  - o ENTITIES AVAILABLE
    - o POINTS
    - o LINES
    - o ARCS
    - o OBJECTS

STRUCTURE DIAGRAM FOR A SEGMENT

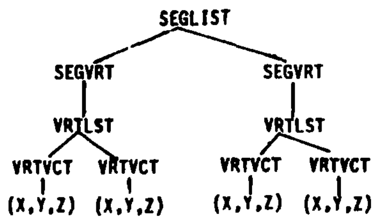


BASE LEVEL LOGICAL

STRUCTURE (SEGMNT)

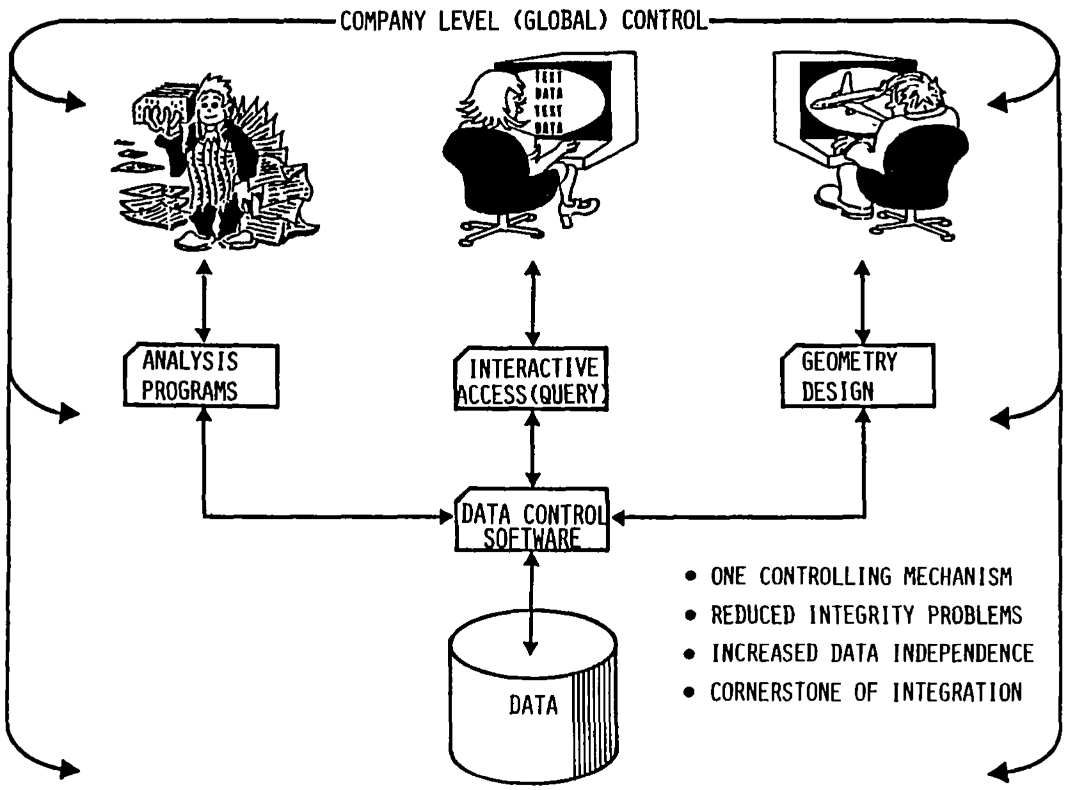
STRUCTURE (VERTEX)

SYSTEM MODEL



USER MODEL

DATA BASE MANAGEMENT ENVIRONMENT



GEOMETRY PROCESSING - FUTURE RESEARCH

- CONTINUE CODE TESTING FOR THE GEOMETRY HIERARCHY
- MATHEMATICAL TRANSFORMATIONS IN THE DBMS
  - ANSI TO IPAD MAP
  - MATRIX MULTIPLY
  - EXTENSIONS TO MAPPING SCHEMA
- SURFACES AND SOLIDS IN IPIP
  - SURFACES AND SOLIDS IN THE DESIGN ENVIRONMENT
  - ESTABLISHES TECHNICALLY FEASIBLE 3-D PART GEOMETRY MANUFACTURING INTERACTION
- DEVELOP INITIAL CONCEPTS OF LOCAL/GLOBAL D.M.
  - GEOMETRY IN BOTH LOCAL AND GLOBAL D.B.
  - GEOMETRY HANDOVER
  - ACCESS AND LOCKING ON DATA COMMON TO LOCAL AND GLOBAL D.B.

BIBLIOGRAPHY

J. W. SOUTHALL, NASA CONTRACTOR REPORT; "DEVELOPMENT OF INTEGRATED PROGRAMS FOR AEROSPACE-VEHICLE DESIGN (IPAD) - INTEGRATED INFORMATION PROCESSING REQUIREMENTS," MARCH, 1979

C. J. DATE, "AN INTRODUCTION TO DATA BASE SYSTEMS," ADDISON-WESLEY, 1977

R. P. DUBE, G. HERRON, J. E. SCHWEITZER, E. R. WARKENTINE, "GEOMETRY DATA MANAGEMENT," IPAD NATIONAL SYMPOSIUM, COLORADO, SEPT. 17-19, 1980

H. R. JOHNSON, D. L. COMFORT, D. B. SHULL, "AN ENGINEERING DATA MANAGEMENT SYSTEM FOR IPAD," IPAD NATIONAL SYMPOSIUM, COLORADO, SEPT. 17-19, 1980

D. TSICHRITZIS AND A. KLUG, "THE ANSI/X3/SPARC DBMS FRAMEWORK"

CAD/CAM DATA MANAGEMENT

Olin H. Bray

Data Management Technology Center
Control Data Corporation
Bloomington, Minnesota

INTRODUCTION

This paper describes the role of database management in CAD/CAM, particularly for geometric data. First, it identifies long-term and short-term objectives for CAD/CAM data management. Second, it explains the benefits of the database management approach. Third, it describes some of the additional work needed in the database area.

OBJECTIVES

The long-term data management objective is an integrated CAD/CAM database. Individual applications, such as design, drafting, analysis, process planning, and numerical control, should extract the data they need from the database and store their results back in the database for use by other applications. Some of this data will be geometric (e.g., curves, surfaces, and solids), but some will be non-geometric (e.g., material properties, process planning data, and product structure information). There are many linkages between geometric and non-geometric data and these relationships can be best defined, maintained, and expanded with a database management system. This integrated database approach will develop relatively slowly because of the complexity of the database and because of the inertia created by the existing applications, most of which are file rather than database oriented.

In the interim a short-term objective is to provide more automated interfaces between existing applications and those which are being developed. Two types of interfaces are possible. The first type is a common interface between applications performing the same function (e.g., an IGES interface for drafting). This common interface minimizes the number of translators and should be "relatively" simple because applications performing the same function use essentially the same data. The second type of interface involves linking applications performing different functions, such as design to drafting or process planning. This type of interface can be much more difficult because it may need to obtain and relate data from several sources and interface several applications simultaneously. A variation of this approach is to develop a common interface database that can be used by many applications, each with its own translators. However, this interface database approach with translators for existing applications, each of which keeps its own internal data structure, is very different from the integrated CAD/CAM database - - the long-term objective.

DATABASE CONCEPTS

Almost all of the CAD/CAM systems today, especially those dealing with geometry, are based on file rather than database management concepts. Database management involves more than simply calling a set of files a database. Database management implies as a minimum a separate database definition (the schema) which specifies both the logical structure of the database and the way it is physically stored and accessed. It also requires support software (the DBMS) which uses that definition to manage the database. Ideally, it also includes a high level query language. Finally, the DBMS should be separate and independent of the applications which use the database.

The DBMS approach has four major benefits. First, the programmer or the end user can concentrate on the function to be performed, not the mechanics of accessing the data, because the DBMS uses the schema to determine the best way to access the data. Second, data integrity is improved because anything specified in the schema can be checked by the DBMS before a change is made or new data is entered. This also speeds up application development because the extensive editing and error checking usually included in the application code can now be done automatically by the DBMS. Third, the schema can include access controls and security checks to be made before a user is allowed to access or modify specific record types or fields in the database. Finally, the database and its applications are easier to maintain. When applications are modified or added, the database and its definition may be changed. With a file oriented system this would require extensive maintenance, i.e., the modification of any applications using the changed files. With database management, however, as long as the DBMS can find a way to map the data from the form in which it is stored into the form the application expects, an application does not have to be changed.

FUTURE WORK

Additional work is needed in both the geometry and the database area, but this paper focuses on the additional work needed in the database management area. Conventional database management systems need to be extended in both the data definition and the query language area, but prototype work can be done with existing DBMS. This prototyping involves defining the logical and physical structure of the database and determining how the applications would interface with the database.

The geometric part of the database can be built up from three basic types of entities -- scalars, points, and vectors. The database definition includes the attributes of these entities and the ways in which they are related. These basic entity types can then be used to build up more complex geometric objects, such as curves, surfaces, and solids. The next step is to relate non-geometric data to specific geometric entities, for example, assigning material properties to solid objects or specifying surface finishes.

Some systems have taken a first step by allowing a user to arbitrarily relate attributes to a geometric entity, but with no separate database definition there is no way to ensure the validity or completeness of these attributes. Also because this data is added to a file used by one application it is not directly available to other applications. Most existing DBMS can resolve this problem.

A system like RIM (Relational Information Manager) has gone a step further. It allows the user to define and use special engineering data types, such as complex, vector, and matrix. The general requirement would be for an extensible data definition language so that a database administrator could define new data types as necessary.

Engineering extensions are also needed in the query language. If engineering data types can be defined, it is reasonable to perform the appropriate operations on them, such as a dot or cross product on vectors.

OUTLINE

CAD/CAM DATA MANAGEMENT OBJECTIVES

DATABASE MANAGEMENT CONCEPTS

FUTURE DBMS ENHANCEMENTS

LONG-TERM OBJECTIVE

INTEGRATED CAD/CAM DATABASE

INCLUDING BOTH GEOMETRIC

AND NON-GEOMETRIC DATA

GEOMETRIC DATA

POINTS

VECTORS

CURVES

SURFACES

SOLIDS

NON-GEOMETRIC DATA

PART NUMBERS

MATERIAL PROPERTIES

SETUP AND RUN TIMES

BILL OF MATERIALS

SHORT-TERM OBJECTIVE

AUTOMATED INTERFACES BETWEEN

APPLICATIONS IN THE

DESIGN AND MANUFACTURING

PROCESS

INTERFACES BETWEEN

SAME APPLICATIONS

DIFFERENT APPLICATIONS

FILE SYSTEM PROBLEMS

INTEGRITY

CONSISTENCY

MAINTAINABILITY

DATABASE MANAGEMENT OBJECTIVES

EASE OF USE

EASE OF MAINTENANCE

INTEGRITY

SECURITY

DATABASE MANAGEMENT COMPONENTS

DATABASE DEFINITION (SCHEMA)

DATABASE

USER - SCHEMA (SUBSCHEMA)

DBMS

FUTURE ENHANCEMENTS

DATA DEFINITION

QUERY LANGUAGE

The Virtual Front End:

Towards Better Management of Solid Geometric Data

William W. Charlesworth
and
Michael J. Bailey

CADLAB, School of Mechanical Engineering
Purdue University
West Lafayette, IN

Introduction

Throughout the last decade, Computer Aided Design has been seen as a Great Promise. Much has been accomplished, but that pales in comparison with what was promised. Ultimate achievement has always seemed "just a step away." During the era of the early turnkey system, the "one step" was limited by modeling geometry through a wire frame. Besides being cumbersome to deal with in three dimensions, it has been continually shown that a wire frame can be ambiguous. This made the modeling of geometry for engineering purposes even more frustrating. The fundamentals of computer aided engineering design rest on the ability to do various analyses. Traditionally, the ambiguity of wire-frame modeling systems has impeded analysis by requiring excessive operator intervention.

Presently, solid modeling is taking the mechanical engineering CAD world by storm. There are no fewer than 20 solid modeling packages around the world, and more appearing with each vendor show. We are still faced, however, with the question of integration, that is, the combination of elements of the CAD process into unified and easily usable design tools.

The CAD Environment Triad

A view of the integrated CAD system that is emerging is of a triad of elements, all three of which are equally important (Figure 1). One of these elements is the set of application programs, the many different data engines that transform data and maintain the validity of the model. This element of the triad has received the most attention. A typical operation found here might evaluate the intersection between sets of primitive objects. This analysis is the basis for many solid modelers. Hidden surface displays are another popular application. Other applications apply the laws of dynamics, strength of materials, etc., to analyze the resulting objects.

Another component of the CAD system triad is the data base. The data base provides the application data engines with input and collects their output. Any information concerning the engineering model is to be found here.

The third component is the user interface. This element sees to it that the designer and the computer stay good friends by reconciling their differences.

The latter two parts of the triad have been badly neglected. Both are usually custom built for and subordinate to a specific application program. The consequence is that the application cannot get its input directly from another application, nor send its results directly to others. The user must intervene to translate the data at both ends, but every time he steps in he risks contaminating the model with errors. The drudgery of entering the same model several times, each time in a slightly different form via a slightly different user interface, may encourage the user to abandon the CAD system for more familiar, traditional design methods.

The obvious solution is to integrate the application programs so they can cooperate with each other without annoying the user. One approach to integration is to create a user interface and a data-base system that support all of the applications but which are independent of the applications. The application programs can then exist independently of each other and still communicate through the data base. It should be possible to add an application not conceived at the time the system was created without affecting the older applications.

The Virtual Front End

This work focuses on the user interface, the front end to the system. Because it is the interface for many applications which do not necessarily exist yet, we call it a virtual front end to distinguish it from the specific interface for a single application. Among the applications are a representative mix of solid modelers, i.e., Romulus, the ICEM Modeler, and TIPS-1, and other programs such as the mechanical dynamics program ADAMS. An Evans and Sutherland PS300, which does dynamic, three-dimensional graphics, is the centerpiece of our work station. A Control Data

CYBER 170-720, a mainframe computer, is our "super minicomputer" representing the computing power that will one day be dedicated to a CAD work station.

A CAD system will not be used if it is not easy to use. "Easy to use" is a nebulous term but a suitable measure might be that the faster information flows between engineer and machine the easier the system is to use. People handle a few abstractions more easily than a host of specifics. Computers lean the other way: they thrive on reams of precise data. Our interface between the two takes advantage of this difference. The human can be explicit about a little, while implying a great deal. For example, he can turn a dial to tell the machine to turn an object. The machine can talk back to the man with a moving picture, so that he can understand immediately all of the transformations the machine has made in the model in order to turn it.

The future designer will be computer literate, writing some of his own applications as well as using applications written by others. This adds another dimension to the "easy to use" system: it must be expandable to do things unimagined at the time it was designed. At the user interface the engineer can combine simpler functions into higher level functions by way of a macro facility. On a deeper level within the system the user can add new primitive functions, and interactively incorporate new commands into the menu library. This requires a mechanism for invoking both the data-base management system and the user interface management system without the need to understand the inner workings of either: there is a clear distinction between the duties of the data base, the front end, and the application.

The principal method of describing geometry through the front end is by way of constructive solid geometry. Simpler objects can be combined to form more complex ones. Figure 2 shows a shifter with all of its primitive objects in place before evaluation by a solid modeler. Figure 3 shows the same object after passing through a solid modeler.

With data on the solid geometry available, further analyses can be done on the model. Tool paths for machining can be generated (Figure 4). In some cases an image of sufficient realism can replace a prototype [1] (Figure 5). Assemblies can be modeled as well as single parts (Figure 6) and further studies, such as a three-dimensional dynamic analysis [2], can be performed on the model.

REFERENCES

[1] Davis, J.E., Bailey, M.J., Anderson, D.C., "Projecting Realistic Images of Geometric Solids," Computers in Mechanical Engineering, Volume 1, Number 1, August 1982, pp6-13.

[2] Rush, Candace, "The Mechanical Simulation of Solidly-Modeled Assemblies," Masters Thesis, Purdue University, West Lafayette, IN, USA, May 1983.

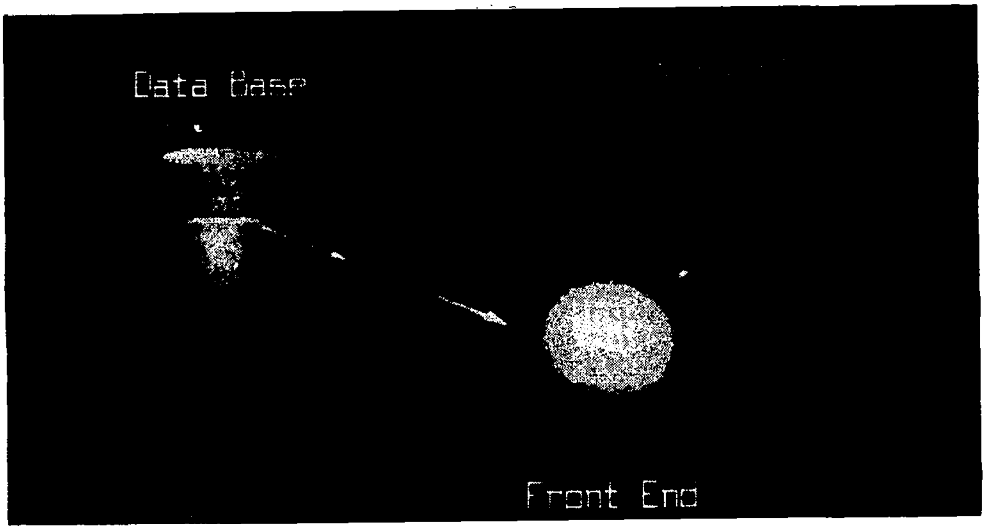


Figure 1.- The CAD environment triad.

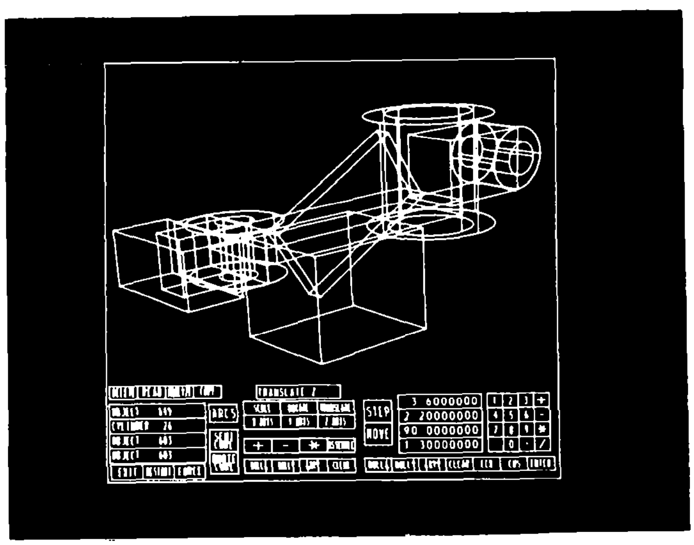


Figure 2.- Shifter modeled on the virtual front end with all primitives shown.

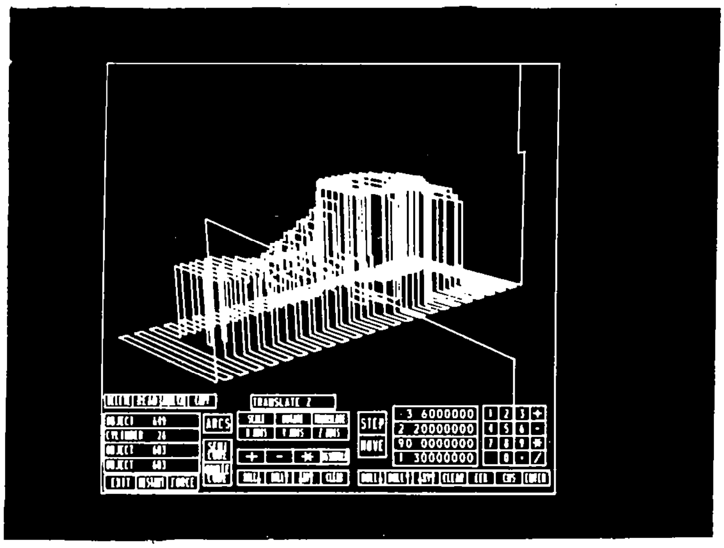


Figure 3.- Shifter evaluated by a solid modeler.

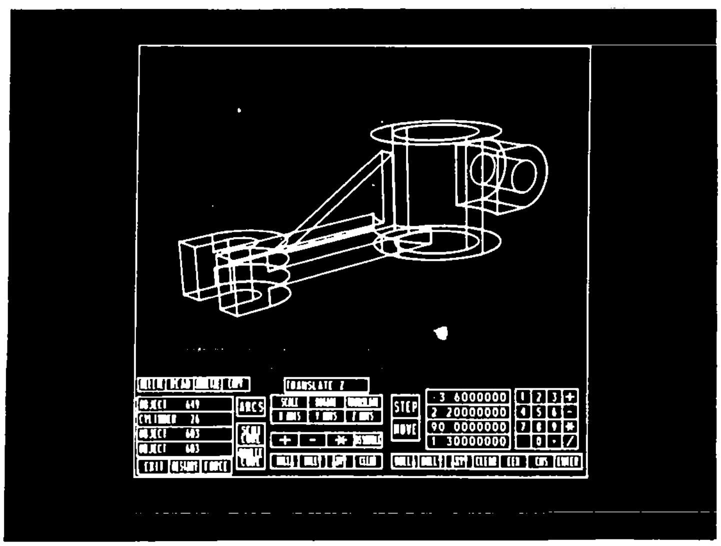


Figure 4.- Final cut tool path for the shifter.

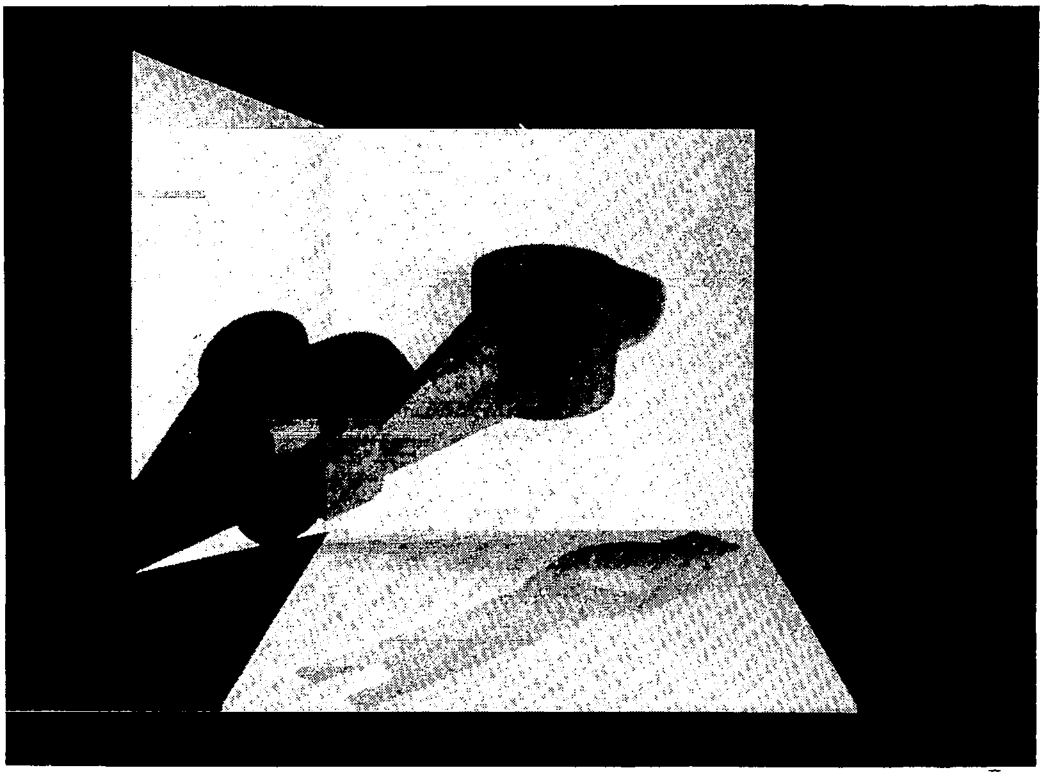


Figure 5.- Realistic image of the shifter.

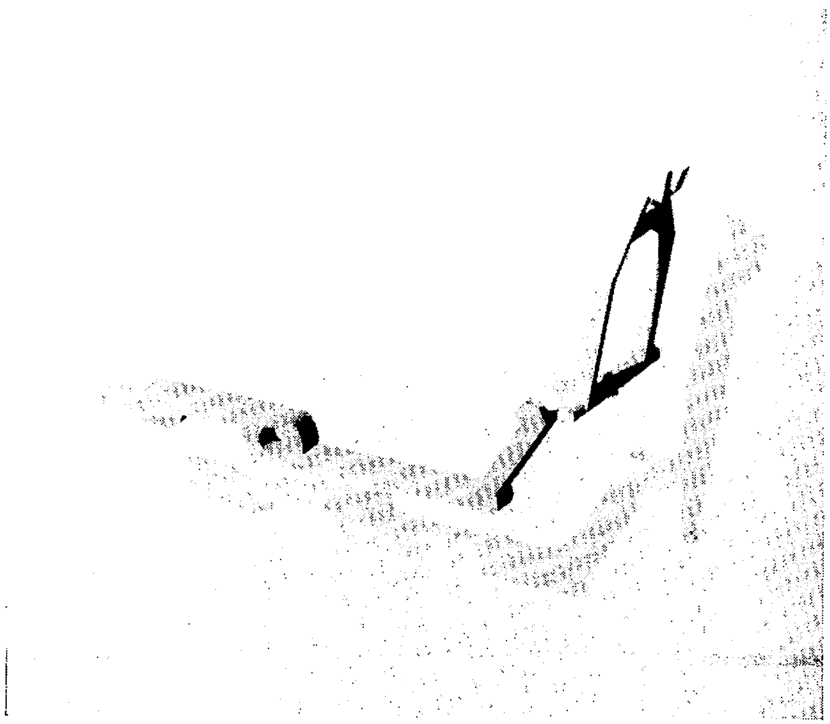


Figure 6.- Hammer mechanism modeled on the virtual front end.

AN ENHANCED OCT-TREE DATA STRUCTURE AND OPERATIONS FOR SOLID MODELING

K. Fujimura,* H. Toriya,** K. Yamaguchi,* and T. L. Kunii*

*Department of Information Science,
Faculty of Science, The University of Tokyo,
Tokyo, Japan

**Software Development Center, Ricoh Co., Ltd.,
Tokyo, Japan

Oct-trees are enhanced to increase the processing efficiency of geometric operations for interactive CAD use. Further enhancement is made to combine them with surface models for more precise boundary specification as needed by tool path generation in CAM applications.

Problems of Existing Data Structures

Internal data structure is essential for 3-D geometric modeling in a computer. Among various representations that have been proposed, constructive solid geometry (CSG) representation and boundary representation are presently widely accepted (refs. 1 and 2); however, in any of these representations, object manipulations such as set operations or display operations require a very large amount of computation. The major part of the computation is used to determine whether a specific point is included in some object or not.

Another problem arises when storing data in a data base. In a data base, it is an essential requirement to ensure the consistency of stored data; for example, physical 3-D interference of mechanical parts must be avoided. To check the consistency of the stored data in these representations requires extensive computation. Generally, multiple representations exist for one object as seen in designing a cutter where the cutter blade needs a few orders of magnitude more precision than the outside cover. This is the area of research that we are studying.

Oct-tree

In order to overcome these problems, an oct-tree representation was proposed several years ago (refs. 3 and 4). The oct-tree is a natural extension of the quad-tree representation (ref. 5) to the 3-D object. Because it approximates objects by various sizes of cubes which are hierarchically organized in a tree structure, it can describe objects with variable resolution levels specified by a user.

In the oct-tree representation, it is an easy task to perform set operations such as union and intersection. Also, because the representation of an object is unique so long as it's position does not change, the equality of objects can be checked easily, a desirable property when building a data base.

Problems of the Oct-tree: Geometric Operations

Unfortunately, little work has been done on geometric transformation (translation, rotation) in the oct-tree representation. Since geometric transformations are operations used most frequently, there exists a need for efficient geometric transformation algorithms. Because a transformation in the oct-tree representation is a combinatorial problem, the manipulation sequence of an oct-tree affects calculation time very much. In this respect, published methods can hardly be said to be efficient. We have developed a new algorithm (figure 1) which utilizes the combinatorial property of an oct-tree. In table 1, we compare the translation algorithm by Jackins and Tanimoto (ref. 3) with our new algorithm and see that the processing time is accelerated by a factor of 10 to 100. Other geometric operations such as rotation and scaling are not easily performed in the original oct-tree representation. For example, an object in oct-tree representation is not rotation symmetric. Hence, after a series of rotations, the object boundary becomes fuzzy. In order to overcome this difficulty, we are now combining the oct-tree representation and the surface representation.

Display Operation of an Oct-tree

Because a display operation is used very frequently, a quick display function is desirable. Several algorithms have been proposed so far, for example, to convert an oct-tree into a quad-tree (ref. 4) and to display octants by the farthest first method (ref. 6). However, the published algorithms are not very efficient and there is much room for improvement.

As to converting an oct-tree into a quad-tree, we found a more efficient algorithm which is approximately two times faster than the published one (ref. 4). The difference between the original algorithm and the new one is the sequence of traversing an oct-tree. The former algorithm traverses an octant from the side farther from the viewer, while our new algorithm traverses an octant from the nearer side. Because octants behind another octant may not be seen by a viewer, they need not be traversed. Thus in our algorithm, we can significantly save traversing time.

An isometric view is very important in engineering designs. To obtain an isometric view, we developed a faster algorithm which utilizes a triangular quad-tree (figures 2 and 3). This improves the display time by at least a factor of three compared with the conventional algorithms (ref. 4). By utilizing the triangular quad-tree, the drawing time may be further shortened by optimizing the display commands at an intelligent graphic display. The essential idea here is to convert the triangular quad-tree into chain codes.

Problems of the Oct-tree: Jagged Surfaces

The oct-tree is an approximation of an object with smooth free surfaces by small cubes. It is intrinsic to this method that the encoded object entails some notched surfaces.

In order to avoid displaying these jagged surfaces, it is necessary to represent an object by a very deep oct-tree. Then, a complex object in considerably high display resolution requires very large data storage and a high speed processor. This means that the most important advantages of the oct-tree such as the processing speed and memory economy are lost.

Solution

If we use the pure oct-tree structure for geometrical modeling, we are soon faced with this problem. We are now building a new system that gives an answer to this problem, while keeping the advantages of an oct-tree.

The basic idea of this new method is to combine the oct-tree representation with a surface model (figure 4). Most of the operations can be performed on the oct-tree structure, while operations such as rotation by an arbitrary angle or display operations can be done with the surface function associated with the oct-tree structure. In this representation, every detail of the object is preserved. Thus, this operation enables one to rotate an object by an arbitrary angle without losing details of an object, an impossible operation in the original oct-tree structure. When we use this functional representation, users can decide the trade-off between storage and resolution level. The set operations for this representation are already implemented. Other operations are now under development.

REFERENCES

1. Requicha, A. G.: Representation for Rigid Solids, Theory, Methods and Systems: Computing Surveys, 12, vol. 4., December 1980.

2. Requicha, A. G.; and Voelcker, H. B.: Solid Modeling, A Historical Summary and Contemporary Assessment: IEEE CG & A, March 1982.

3. Jackins, C. L.; and Tanimoto, Steven L.: Oct-Trees and Their Use in Representing Three-Dimensional Objects: Computer Graphics and Image Processing, vol. 14., pp. 249-270, 1980.

4. Doctor, L. J.; and Terborg, J. G.: Display Techniques for Octree-Encoded Object: IEEE CG & A, July 1981.

5. Alexandridis, N.; and Klinger, A.: Picture Decomposition Tree Data Structures, and Identifying Direction Symmetries as Node Combinations: Computer Graphics and Image Processing, vol. 8., pp. 43-77, 1978.

6. Meager, D.: Geometric Modeling Using Octree Encoding: Computer Graphics and Image Processing, vol. 19., 1982.

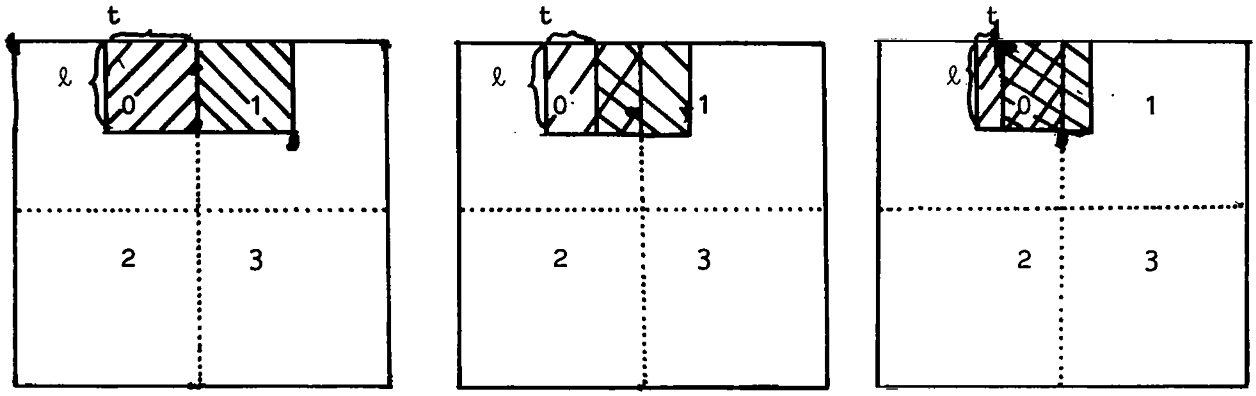
TABLE 1.- COMPARISON BETWEEN OUR TRANSLATION AND TANIMOTO'S TRANSLATION

Node number				Translation distance	Execution time (second)		
Original node		Resultant node					
Node	Leaf	Node	Leaf	(x , y , z)	(1)	(2)	(3)
1	1	1	1	(512, 0, 0)	0:11	0:01	0:01
1	1	3	8	(256, 0, 0)	0:13	0:01	0:01
1	1	11	36	(128, 0, 0)	0:30	0:01	0:07
1	1	43	148	(64, 0, 0)	2:01	0:03	0:48
1	1	171	596	(32, 0, 0)	8:25	0:23	7:41
56	8	52	8	(512, 0, 0)	1:21	0:10	0:11
56	8	32	8	(496, 0, 0)	10:26	0:11	0:12
56	8	36	8	(480, 0, 0)	7:44	0:08	0:08
56	8	40	8	(448, 0, 0)	5:30	0:12	0:12
56	8	44	8	(384, 0, 0)	3:40	0:11	0:12
56	8	48	8	(256, 0, 0)	2:30	0:11	0:12
56	8	44	8	(128, 0, 0)	3:27	0:11	0:14
56	8	40	8	(64, 0, 0)	4:36	0:11	0:13
56	8	36	8	(32, 0, 0)	6:10	0:11	0:14
56	8	32	8	(16, 0, 0)	7:44	0:11	0:13
56	8	28	8	(8, 0, 0)	9:34	0:10	0:12
56	8	64	36	(4, 0, 0)	11:32	0:15	0:19
56	8	14	8	(120,120, 0)	79:02	0:18	0:10
56	8	14	8	(56, 56, 0)	71:03	0:16	0:11
56	8	38	8	(128,128, 0)	5:32	0:17	0:12
56	8	32	8	(64, 64, 0)	10:01	0:19	0:15
56	8	26	8	(32, 32, 0)	15:15	0:16	0:11
56	8	20	8	(16, 16, 0)	24:00	0:17	0:11
56	8	14	8	(8, 8, 0)	35:10	0:17	0:10
56	8	72	50	(4, 4, 0)	59:07	0:26	0:20
56	8	42	8	(256,256,256)	5:05	0:31	0:12
56	8	35	8	(128,128,128)	12:10	0:28	0:12
56	8	28	8	(64, 64, 64)	32:02	0:26	0:12

(1) Translation by Tanimoto.
(2) Translation with gap (our new method).
(3) Translation without gap (our new method).

A cube in an oct-tree is specified by a node address such as 0-1.

t : the distance of translation.
ℓ : the length of a target cube's edge.



(1) t=ℓ (2) t = ℓ/2 (3) t = ℓ/4

original node address ---(translation)---> new node address pattern

original node address	(1)	(2)	(3)
0-1	1-0	0-1-S1 1-0-S0	0-1-S1 -S0-S1 1-0-S0-S0

Notation : S0 = { 0,2,4,6 }
S1 = { 1,3,5,7 }

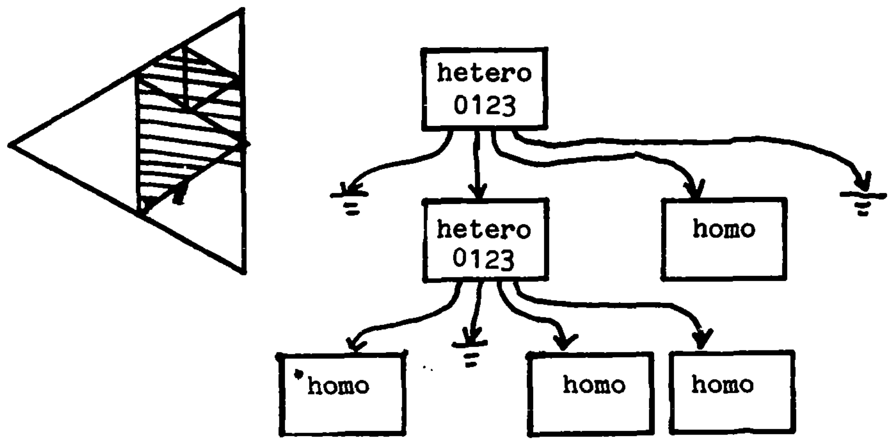
0-S0 means 0-0
-2
-4
-6

0-S1 means 0-1
-3
-5
-7

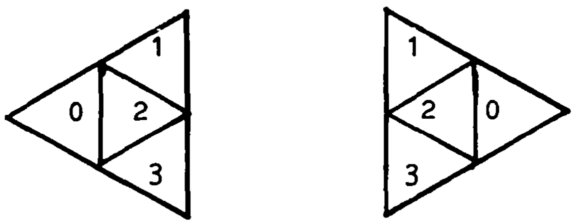
S0-S0 means 0-S0
2-S0
4-S0
6-S0

If t is a multiple of ℓ, the translation node address is obtained by the permutation of the original node address. Otherwise, new node addresses are obtained from the original node address by the above translation rules.

Figure 1.- Basic concept of gap translation.



Triangular Quadtree and its corresponding area.

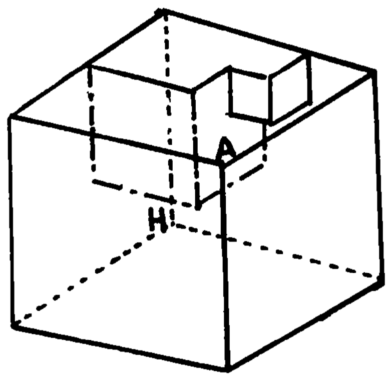


Area numbering conventions.

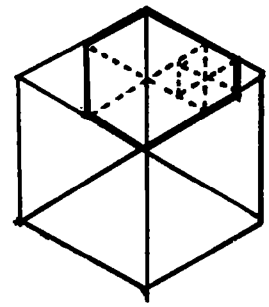
Triangular area is divided into 4 triangles, when the area is heterogeneous.

Figure 2.- Triangular quadtree.

When we see a cube (1) in an isometric direction, the view point, vertices A and H are aligned, and its projection becomes a regular hexagon.



(1) Sample object



(2) Isometric view

3-D area (1) corresponds to (2) in the isometric view. (2) can be represented by the triangular quadtree as shown below.

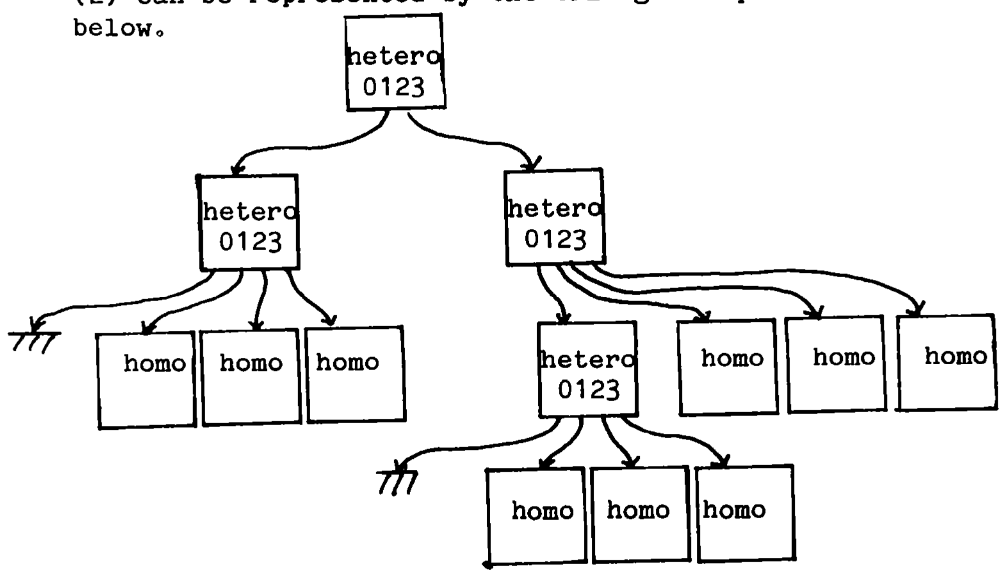
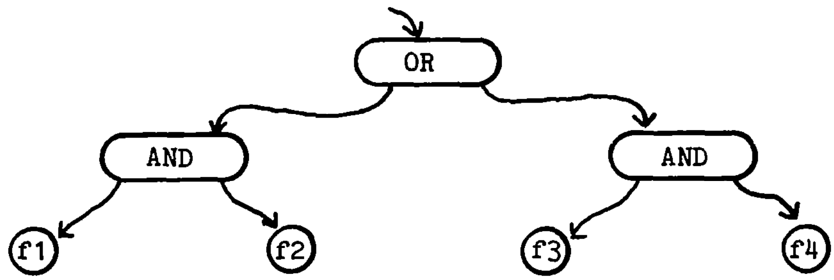


Figure 3.- Isometric view and the triangular quadtree.

Suppose an object is defined by

(f1 > 0 and f2 > 0) or (f3 > 0 and f4 > 0)

We describe it by the following tree called a "function tree".



Then, an oct-tree with function trees is described as follows.

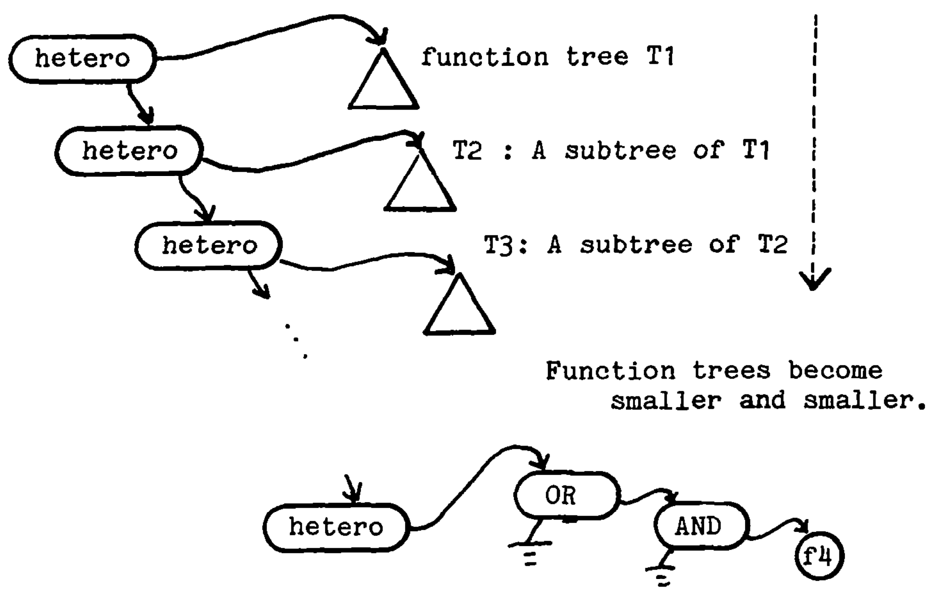
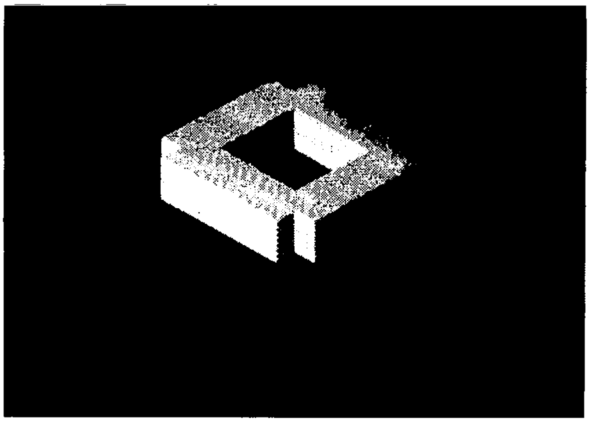
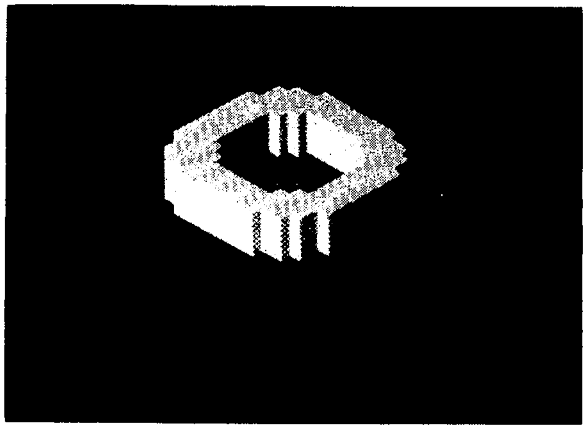


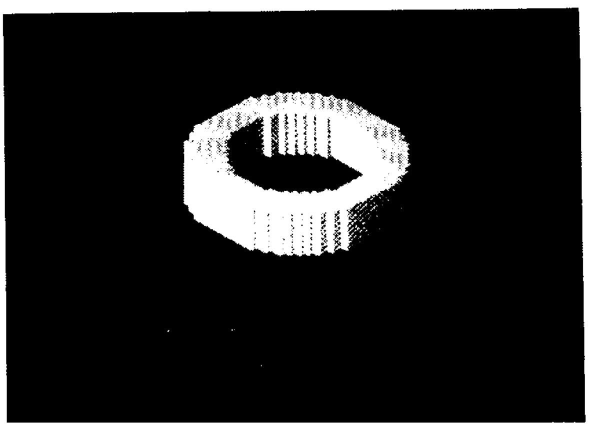
Figure 4.- Combining the oct-tree representation with a surface model.



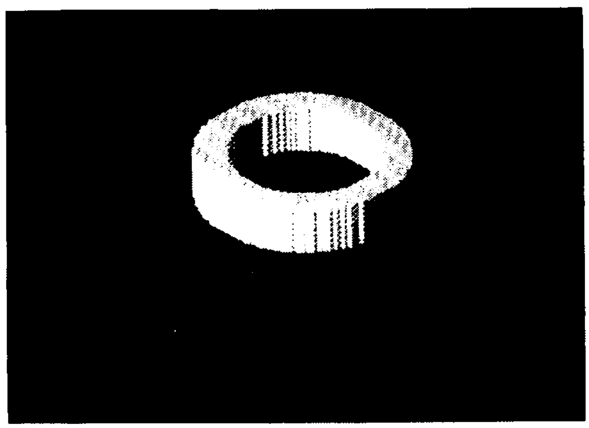
Example 1



Example 2



Example 3



Example 4

Multiresolution Examples

IGES, A Key Interface Specification for CAD/CAM Systems Integration

Bradford M. Smith
Joan Wellington
National Bureau of Standards

ABSTRACT

The Initial Graphics Exchange Specification (IGES) program has focused the efforts of 52 companies on the development and documentation of a means of graphics database exchange among present day CAD/CAM systems. The project's brief history has seen the evolution of the Specification into preliminary industrial usage marked by public demonstrations of vendor capability, mandatory requests in procurement actions, and a formalization into an American National Standard in September 1981 [1]. Recent events have demonstrated intersystem data exchange among seven vendor systems with a total of 30 vendors committing to offer IGES capability. A full range of documentation supports the IGES project and the recently approved IGES Version 2.0 of the Specification [2].

Today all industrialized nations of the world are being challenged to increase productivity in the design and manufacture of products. At the same time, they must face problems of increased product complexity and shortened product life cycles. The development and growth of computer-aided design and manufacturing, commonly known as CAD/CAM, provided a partial solution to these productivity problems.

However, as more and more users turned to CAD/CAM equipment to increase their productivity, they realized that the full potential of this equipment could not be met without a method for communicating data between different systems.

In September 1979 representatives from government and industry joined forces under the Air Force ICAM program to develop this method for data exchange. Funding for management and coordination was provided by the Army, Navy, Air Force, and NASA through the ICAM program. Industrial users and CAD/CAM system suppliers provided resource material and personnel.

Development of the data exchange method was assigned to a technical committee with representatives from the National Bureau of Standards (NBS), the General Electric Company, and the Boeing Company, with coordination for the overall effort assigned to NBS. The result of this industry-wide effort was the creation of the Initial Graphics Exchange Specification, known as IGES, which was published in January 1980 as an NBS report [3].

Just what is IGES and how can it increase productivity for your organization? IGES is a data format for describing product design and manufacturing information which has been created and stored in a CAD/CAM system in computer-readable form. The IGES format is in the public domain and is designed to be independent of all CAD/CAM systems.

The benefit of this common format is that a user does not have to develop special translators for each different piece of equipment that is used. The only requirement is to have a translator to and from the IGES format. These translators, called pre- and post-processors, are generally available from the equipment vendor. In addition, an IGES file can be stored on magnetic tape or disk memory for future use. It can also be transmitted between systems via telecommunications.

STANDARDIZATION STATUS

The first major step toward standardization under the procedures of the American National Standards Institute (ANSI) occurred in May of 1980. At that time the ANSI Subcommittee Y14.26 on Computer Aided Preparation of Product Definition Data voted to include the IGES Version 1.0 document as Sections 2 through 4 of a five-part draft standard to be submitted into the formal review procedure.

As a part of that effort the IGES document was reformatted into the form required for ANSI release. In addition, an introductory Section 1 was written and Sections 2 through 4 were integrated with Section 5, which provides a more sophisticated mathematical representation of three dimensional objects. Section 5 was developed by the Y14.26 committee.

In October 1980 the draft standard was distributed for ballot and comment. In actuality, three separate reviews took place at that time: the document was provided to members of the Y14 Committee on Drafting Practices and to members of the Y14.26 Subcommittee, and was released to the general public. All were asked to ballot on the proposed standard.

Resolving the hundreds of technical comments resulting from the balloting was a formidable task, but progress was speedy, thanks to the superior efforts of the IGES ANSI Response Task Group. Many long hours were spent by task group members pursuing two equally important objectives of the task. The first was the analysis of all comments received and the incorporation of positive contributions into the draft document to make it technically more sound and precise. The second, of course, was to achieve a consensus on the draft standard by resolving those comments which accompanied negative ballots. As an indication of the magnitude of this effort, some 384 individual technical points were anlayzed with 280 incorporated into the draft document as editorial changes, changes for consistency or clarifications to the technical intent. Final adoption by ANSI of the Y14.26M document was achieved in September 1981. This was just 17 months after initial submission.

TRANSLATOR DEVELOPMENT

The goal of the IGES project is to achieve portability of data among various CAD/CAM systems. Certainly the development of a national standard is a major step toward that goal. But portability will not be realized until quality translator implementations are in widespread use. Recent events have contributed much toward this goal from both a user and a vendor standpoint. Users are now actively writing translators to their in-house developed CAD/CAM software packages, and vendors in the graphics community are now supplying or are publicly committing themselves to supply IGES translators. Finally, an increasing amount of testing of IGES translator capability has begun between different graphics systems.

USER IMPLEMENTATIONS

Many users of CAD/CAM systems have invested heavily in the development of special-purpose software for the design, analysis and testing of their discrete products. As they seek to integrate this software into total design and manufacturing systems, many are making use of IGES to solve their problems of database communication. The work has a dual focus: transfer of product definition data within the corporate system, and digital communication between the company and its suppliers and customers. Efforts are known to be under way in eleven different firms. They are: Audi, Germany; Bendix; Boeing; Deere & Co; Ford; General Electric; General Motors; Lockheed; Martin Marietta; SDRC; and Westinghouse.

VENDOR IMPLEMENTATIONS

From the inception of the IGES project, the graphics vendor community has provided good technical support toward its development. As early as September 1980 three firms, Applicon, CALMA, and Computervision, announced initial translator capability at the SME Autofact Conference. In November 1981 at Autofact a fourth firm, Control Data Corporation, joined the list of implementors. At the National Computer Graphics Association (NCGA) Exposition in June 1982 three additional firms, Gerber, MCAUTO-Unigraphics, and Manufacturing and Consulting Services, provided tapes or showed translator capability. In total, thirty vendors are now committed to supplying IGES translators for their products. Figure 1 presents this information on vendor implementations.

INTERSYSTEM TESTING

The first opportunity for exchange of IGES files among different computer systems occurred in the fall of 1981 with the publication of the Test Library [4]. This document and accompanying magnetic tape contain 36 individual test cases of IGES entities. The library has been of great value to implementors in testing the validity of their software and has been a major step toward the development of quality translator software.

In December 1981 the first publicly documented intersystem transfer of IGES information in an actual working environment occurred between two operating facilities of the Department of Energy (DOE). The mechanical part shown in Figure 2 was designed and detailed on a Computervision CADDS 4 system located at Sandia National Laboratories in Livermore, California. Three-dimensional model data describing the geometry of this part was expressed in the IGES format on magnetic tape and transported to the Bendix Corporation in Kansas City, MO. There it was interpreted on the Control Data Corporation CD 2000 system where data was added to define a cutter path for subsequent NC machining. A production print from Sandia was used during final inspection to verify part accuracy. The results of this transfer of product data using IGES are fully documented in a report published by Bendix. The IGES translators used were commercially available, vendor-supplied standard pre- and post-processors.

At this time, the first public test of IGES capabilitv was planned for the next opportunity the graphics vendors would be available in the same location. This was provided in June 1982 at the NCGA Exposition in Anaheim, California.

Several tests were performed during that three-day exposition. Applicon, CALMA, Computervision, Gerber, and Manufacturing and Consulting Services participated in the walkthrough. Two other vendors, Control Data and MCAUTO-Unigraphics, supported the demonstration by providing previously recorded magnetic tapes with sample part geometries in IGES format. The demonstration started with sample model geometry being read into the first vendor's system. The 3-D geometry of the mechanical part from DOE appeared on the screen. Additional geometry was added to the part and the resulting file was written out on an IGES tape. This tape was carried to the next vendor where it was read in and displayed on the screen. The tests proceeded on two different days among the participating vendors. Changes to the model geometry were made at each site and could be seen at all successive locations. In this way, the demonstration vividly showed the excellent progress being made in the vendor implementations.

Some minor problems were noted during the demonstrations but did not detract from the objectives of the test. The most frequent problem was with the magnetic tape drives. The other problem, a misplacement of arcs, was well documented and was fed back to the implementors of the translator software. In total, the visitors to the NCGA Conference were able to witness a very successful first public demonstration of the IGES capability.

The AUTOFACT 4 conference and exposition held in Philadelphia from November 29 to December 2, 1982, provided still another opportunity for testing and public demonstration of IGES capability. Five graphics vendors participated; Computervision, Control Data Corporation, Gerber, IBM, and MCS, Inc.

Preparations for this test of IGES processors started with part geometry developed by the IGES Test, Evaluate & Support Committee. Figure 3 is a screen copy of the original test part containing the following entity information:

six points
six lines
four arcs
one conic
fourteen linear/ordinate dimensions
one angular dimension
one radial dimension
one label

This data describes the full range of dimensioning needed for communicating engineering drawings. The data was taken in turn to each manufacturer's booth, loaded into a CAD/CAM system, displayed, and then recorded again on a new IGES tape to be carried to the next system. This testing is presently continuing with composite tapes containing all of the output from the various vendors currently circulating for further analysis and testing.

THE IGES ORGANIZATION

Response to this exchange format has been outstanding. In the first two years after publication, over 1200 copies of the Specification [3] were provided in answer to requests from industry, the Federal Government and the academic community, and additional requests are received daily. Representatives from over 50 companies including all major vendors of CAD/CAM equipment currently serve on committees which have been established to aid in the implementation of the Specification.

To accomplish this task, the committee structure has been established under the leadership of the National Bureau of Standards. Overall direction and policy is provided by the Steering Committee which is chaired by the member from the National Bureau of Standards. Other Steering Committee members are management personnel from four different interest sectors: military/government, suppliers of CAD/CAM systems, industrial users of CAD/CAM systems and members-at-large.

Reporting to the Steering Committee is the Working Committee. Meetings of the whole of this committee are used for dissemination of information and balloting on work of the subcommittees. The majority of technical work occurs in two permanent subcommittees, the Extensions and Repairs (E&R) Committee and the Test, Evaluate and Support (T,E&S) Committee.

The E&R Committee has primary responsibility for the technical quality of the Specification, and as such deals with all changes and additions. Its subgroups are active in areas of advanced geometry, finite-element modeling, electrical/electronics, piping systems and text fonts. In addition, the E&R Committee is responsible for the work which has recently produced Version 2.0 of IGES [2].

The T,E&S Committee has the primary responsibility of providing the tools to ensure quality translator software development. Assistance provided to implementors consists of technical review of implementations, assistance in resolving problem areas, and the general exchange of information to support the overall implementation of IGES.

One of the first products of this committee is the IGES Test Library mentioned earlier. In addition, it is developing a Recommended Practices Guide to serve as an aid for future implementors by providing descriptions of generally accepted alternatives to common IGES issues. The guide will further serve to establish a general philosophy for IGES implementations. When this committee discovers ambiguous or erroneous areas, it forwards issues which require resolution within the IGES Specification to the E&R Committee.

Additional subcommittees have been formed from time to time to address special tasks such as development of a Management Briefing and a Technical Briefing, and coordination and resolution of ANSI ballot responses.

VERSION 2.0

Version 2.0 of the Initial Graphics Exchange Specification (IGES) represents both a refinement and an extension of the earlier published work [3]. Clarity and precision of the Specification have been dramatically improved as the result of wider public review and comment plus feedback from an ever increasing amount of implementation and testing. In addition, many extensions and enhancements have been incorporated in the Specification to expand its capability to communicate a wider range of product data developed and used by computer-aided design and manufacturing systems. Despite these extensions and enhancements, Version 2.0 remains nearly upward compatible with Version 1.0. The only exception is a change in the Text Font Definition entity. The Version 2.0 document was approved in July 1982 by the IGES committee structure.

The many changes dispersed throughout the Version 2.0 document make it difficult for a reader to compare it with the earlier work. This task is compounded by a substantial change in the basic format of presentation. Hence, it is useful here to elaborate the primary differences that do exist. In addition, the reader of the document is aided by vertical bars in the margin that identify areas of non-trivial change. Of course, a complete record of every change is available from the Extensions and Repairs Committee's formal Change Control System, which documents both the request for a change and the actual text modifications to implement the change.

Changes that will be noted in the Specification can be classified into four general areas: editorial, consistency, clarification, and technical extension. Editorial changes include the usual grammar, spelling, punctuation, etc. that are discovered with each re-reading. Changes to improve the consistency of the document include the use of the same terminology throughout, the establishment of a common format, and the defining of all terms before their use. In addition, a Glossary of Terms and an Index of Topics have been added. These changes are not denoted by change bars.

Users of Version 2.0 of the IGES Specification will be pleased to see the many technical extensions which have been added to augment its capability and expand it into new areas. Many geometry entities have been enhanced in scope to be more generally applicable. Included here are the parameterization in the Ruled Surface entity, a more general form of the Tabulated Cylinder entity, and the means of relating the Surface of Revolution entity to common geometrical surfaces like spheres and cones.

Two new geometry entities, a Rational B-Spline Surface entity and a related Rational B-Spline Curve entity, were added in Version 2.0. The addition of these entities is expected to provide a much more general approach for surface and curve representation. Algorithms were developed for an exact conversion between the Rational B-Spline method and the Bezier method of representation. New structural entities were also developed and documented for both rectangular and circular arrays of geometric entities.

In the annotation area, Version 2.0 improves on the earlier work by specifying a much larger set of text fonts, although additional work remains to be done here. Improvements have been made in the clarity of intent for positioning and scaling of text material and in a more clearly defined Angular Dimension entity.

Two major applications areas have been addressed by Version 2.0: finite-element modeling data and electronics printed wiring board product data. The earlier IGES Specification contained no means of handling this data, yet both are widely used applications on CAD/CAM systems.

The geometry of a finite element is defined by the ordered connection of geometric points called nodes. Communication of this data through IGES Version 2.0 is handled by defining the nodes with the Node entity. The node connectivity is defined by the Finite Element entity. To complete the finite-element definition, new properties will be defined to communicate such items as modulus of elasticity, Poisson's ratio, thermal conductivity, moment of inertia, shear modulus, and physical constraints.

The second major applications area addressed in the extended Specification is the communication of printed wiring board product data. Extensions in this area are intended to provide for transferring the physical shape of metallization on each layer, the location and size of drilled holes, the location and identification of components and their pins, the connectivity of certain component pins and their associated named signals, and the functional use of entities by graphics systems level. Some beginning is made into the transfer of design rules and in transferring schematic drawings, but only the physical design transfer is thought to approach complete coverage.

Altogether, four new geometric entities are provided for efficient transfer of commonly used printed wiring board graphical elements, eight new properties are defined to preserve design characteristics, one new associativity is specified for signal connectivity, and one change is introduced so that an IGES property may apply to levels as well as to individual entities.

A frequent criticism of the IGES format has been the anticipated large file lengths due primarily to the ASCII character representation. Included in Version 2.0 are the details of an optional or alternate binary format representation which addresses the problems of file size and processing speed. While efficiency improvements vary with word length and other variables, analysis of 20 IGES production files has estimated the savings in file size at 50 to 68%.

In total, the IGES Version 2.0 document is a major improvement in the Specification. It refines and more precisely describes the Version 1.0 capability as well as extends IGES into new geometry and application areas. Altogether, the document represents the discussions from 98 Change Requests which generated 56 documented Change Orders from the earlier work.

FOR MORE INFORMATION

A great variety of formal documentation exists to describe the IGES Specification and its application to CAD/CAM processes [1-3,5]. In addition to the Test Library, a MACRO Manual [6] has been prepared to assist those implementing the MACRO section of the Specification. A wide range of documentation is available from the technical committees and working groups, as well. For each, the documentation includes a scope of work for the technical activity, the mailing list of interested parties in that work area, and finally, a list of working documents currently active in that work area. To keep the CAD/CAM community up-to-date on IGES activities, a newsletter is published by the National Bureau of Standards on an as-needed basis.

SUMMARY

Many organizations are anticipating the use of new and improved CAD/CAM systems in an integrated fashion to achieve productivity gains. As you can see, IGES provides a way to achieve that integration. It holds great potential as a common communications format among automated functions in design, engineering analysis, manufacturing, and part inspection. Additionally, it may serve as a vehicle for meaningful communication of product definition data among different companies over the full lifetime of a product.

In the future, additional CAD/CAM applications will be demanded by users. A standardized communications interface will be essential among the various modules of a CAD/CAM system -essential if these systems are to be flexible enough to adapt to changing priorities and essential if users are to realize the full potential of their equipment. IGES provides that interface.

The present Specification is well developed and tested and is further strengthened by the wide range of supporting technical literature, all of which is in the public domain. Its data exchange technique is well supported by the vendor community. While IGES is not perfect yet and does not solve all CAD/CAM data exchange problems, it goes a long way toward solving users' current data exchange problems and has the capability of being extended to meet the needs of this growing and maturing field.

REFERENCES

1. American National Standard Y14.26M, Digital Representation for Communication of Product Definition Data, The American Society of Mechanical Engineers,* 1982 (Order Number: N-000-99).

2. Bradford M. Smith, Kalman M. Brauner, Philip R. Kennicott, Michael Liewald, Joan Wellington, et al, Initial Graphics Exchange Specification (IGES), Version 2.0, NBSIR 82-2631 (AF), National Bureau of Standards, 1982 (NTIS** Order Number PB 83-137448).

3. Roger N. Nagel, Walt W. Braithwaite, Philip R. Kennicott, Initial Graphics Exchange Specification (IGES), Version 1.0, NBSIR 80-1978 (R), National Bureau of Standards, 1980 (superseded by ANSI Standard Y14.26M).

4. Bradford M. Smith, IGES Test, Evaluate, and Support Committee, Michael Liewald, Chairman, Initial Graphics Exchange Specification Test Library Version 1.3, National Bureau of Standards,***1981.

5. J. C. Kelly, Robert Wolf, Philip Kennicott, Roger N. Nagel, Joan Wellington, Editors, A Technical Briefing on the Initial Graphics Exchange Specification (IGES), NBSIR 81-2297, National Bureau of Standards, 1981 (NTIS** Order Number PB 81-238719).

6. Neil Webber, Roger N. Nagel, A MACRO Manual for the Initial Graphics Exchange Specification (IGES) (preliminary draft), National Bureau of Standards,*** 1980.

*American Society of Manufacturing Engineers, 345 East 47th Street, New York, N.Y. 10017.

**National Technical Information Service, 5285 Port Royal Road, Springfield, VA 20161.

***IGES, National Bureau of Standards, Building 220, Room A-353, Washington, D.C. 20234.

IGES VENDOR IMPLEMENTATIONS

	Planning To Implement	Using Test Library	Supplied Tape For Testing	Performed Public Intersystem Testing	Supplied Translators To Customers	Reference Note
APPLICON	Y	Y	Y	Y	2	2
A M BRUNING	Y					4
AUTOTROL	Y					2
BAUSCH & LOMB	Y					4
CADAM INC	Y	N			0	1
CADLINC	Y					2
CALCOMP	Y					4
CALMA	Y	Y	Y	Y	12	1
COMPEDA	Y					2
COMPUTERVISION	Y	Y	Y	Y	50	1
CONTROL DATA	Y	Y	Y	Y		1
DATA TECHNOLOGY	Y					4
ENGINEERING SYSTEMS	Y					4
GERBER	Y	Y	Y	Y	0	1
GRAFTEK	Y				0	1
GRAPHCON	Y					4
HOLQUIN	Y					4
HONEYWELL	Y					2
IBM	Y	Y	Y	Y		5
INTERGRAPH	Y	Y				2
K & E	Y					2
MCAUTO UNIGRAPHICS	Y	Y	Y	Y		2
MCS I	Y		Y	Y	48	2
NCS	Y					4
PRIME	Y					2
PROJECT SOFTWARE	Y					4
SUMMAGRAPHICS	Y					4
SYSTEMHOUSE	Y					4
TEKTRONIX	Y					2
T & W SYSTEMS	Y					4

REFERENCE NOTE

1. IGES Questionnaire 5/82
2. Telephone Contact
3. Third-Party Source
4. Machover Survey 4/82
5. Press Release

Figure 1

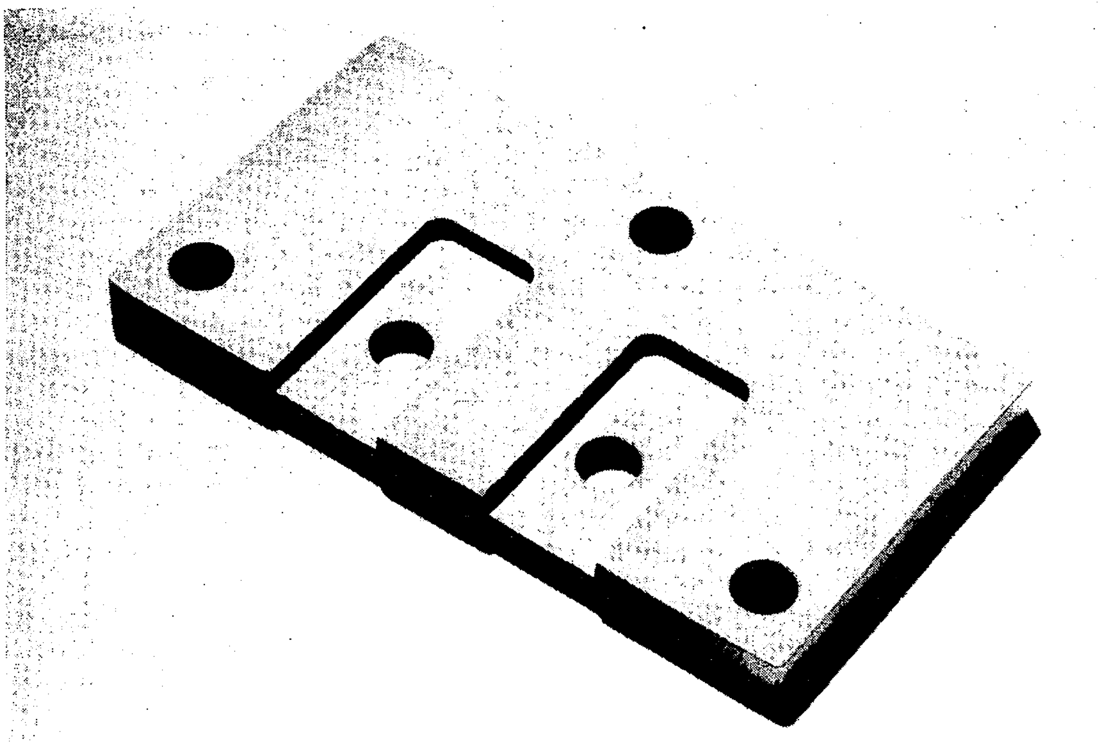


Figure 2

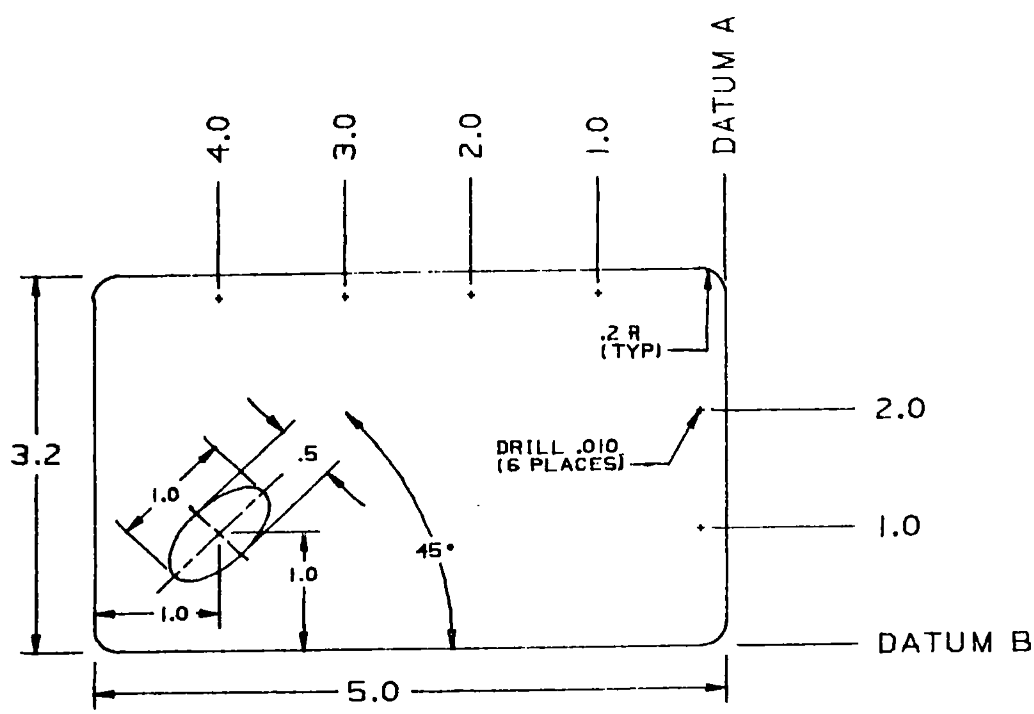


Figure 3

REFLECTIONS ON REPRESENTING NON-GEOMETRIC DATA

R. F. Emnett, McDonnell Douglas Automation Company, St. Louis, Missouri
H. H. Shu, McDonnell Douglas Astronautics Company, St. Louis, Missouri

INTRODUCTION

The American National Standard Y14.26M-1981 on Digital Representation for Communication of Product Definition Data includes an introduction, three sections corresponding to IGES (Initial Graphics Exchange Specification) Version 1.0, and Section 5, which is a constructive, relational, language-based representation for geometric and topological entities (ref. 1).

ANSI5 and ANSI6

Section 5 of the standard (ANSI5) (ref. 1) provides a syntax for an extensible language and defines semantics for geometric and topological structures which may be used to unambiguously represent basic shape data.

The Foreword of the Standard indicates that a Section 6 is to be developed which will address "non-geometric data for three-dimensional object models." This presentation does not represent extensive work in the area; rather, it supplies some reflections on how a Section 6 (herein, ANSI6) might take advantage of the ANSI5 language framework.

Section 1 of the Standard provides a categorization of product definition data by their principal role in defining a product:

- o Administrative (product identification, product structure)
- o Design/Analysis (idealized models)
- o Basic Shape (geometric, topological)
- o Augmenting Physical Characteristics (dimension and tolerance, intrinsic properties)
- o Processing Information
- o Presentational Information

ANSI5 addresses only basic shape data. Presumably, the remaining categories are the province of ANSI6.

TWO EXTENSIONS

The remaining categories cover a wide range of data, from the strictly descriptive to alternate (idealized) geometry. The language syntax described in ANSI5 can be extended to these areas. Two suggested extensions, intended to stimulate discussion, are add-on attributes and new structure types. Examples are presented to demonstrate these concepts.

A non-geometric datum applies either to a single entity or to a set of entities. The datum is either an attribute of an entity or describes a relationship between entities. The line font (solid, dashed, etc.) in which a curve is displayed, and the display of dimensioning and tolerancing data are examples of the two cases. Add-on methods are suggested for the first case, and new structure types for the second.

ANSI5 LANGUAGE REVIEW

Figure 1 depicts the organization of the ANSI5 structures. A brief review of the ANSI5 language syntax by example follows. Consider the language statement and the corresponding (relational) table expression in Figure 2.

LINEX identifies the language statement. LIN identifies it as a linear structure. ENT0 and ENT1 are explained in ANSI5's linear structure definition as entities which are to be interpolated linearly to define a new entity. In ANSI5, these structure constituents are specified for each structure. POINT1 is the domain (in this case, another language statement) in which the entities P1, P2, P3, and P4 are defined and L1, L2, and L3 are the entities defined in this statement; for example, L1 is a linear blend between P1 and P2.

A second ANSI5 language syntax, used for domain definitions, was provided mainly for extension into areas of non-geometric data. An example in both language and tabular form is found in Figure 3.

Here, SHADE identifies the statement, DOM identifies a domain structure, and 5*CHAR indicates that the entries ('GREEN', etc.) are character data with a maximum of five characters each. OPEN indicates that the list of entries may not be complete.

The above synopsis will not provide an in-depth understanding of the two ANSI5 language formats, but should allow the reader to follow concepts presented below.

ADD-ON ATTRIBUTES

For non-geometric data applicable to individual entities, adding a structure constituent to any of the applicable ANSI5 basic shape structures is suggested. The initial example of line font style will generally apply to more than one entity. However, it is a property of each entity individually rather than a relationship among the entities. While grouping methods may be desirable for this situation, such are not addressed herein.

For the add-on attribute, ANSI6 should provide:

- o structure constituent, to identify the non-geometric datum
- o domain for the structure constituent
- o semantics for the structure constituent and domain

NEW STRUCTURES TYPES

Some non-geometric data are not well suited to the add-on methodology. Display of various dimension and tolerancing information is an example. For these cases, ANSI6 might define the same elements as ANSI5 does for basic shape data:

- o structure code
- o structure constituents
- o semantics including applicability and the acceptable values for the entity constituents

EXAMPLES

A simple bolt might be depicted in a two-dimensional drawing as in Figure 4 or simplified to Figure 5. Figure 6 comprises the ANSI5 language statements to communicate the geometry of the latter drawing, and Figure 7 focuses on the point and linear structure statements with their tabular representations.

For the fonts representing curves on graphics displays, the add-on attribute LFONT is suggested. LFONT is the structure constituent. FONTC (Figure 8) is the domain.

ANSI6 should provide semantics for the domain of the structure constituent. Users expanding the domains (such domains should be 'OPEN') are responsible for additional semantics essentially as private convention.

Semantics for the structure constituent LFONT might specify that curve representation default to a solid font. BLANK would be non-displayed curves. DASH, CENTER, and PHANTOM fonts might be:

```
DASH     - - - - - - - - - - - - - - - - -
CENTER   __ _____ __ _____ __ _____ __
PHANTOM  _____ __ __ _____ __ __ _____
```

A parallel methodology for point fonts might use domain FONTP (Figure 8) with the semantics defining 'BLANK' as the default display.

Figure 9 shows in tabular representation the change to the relations POINT2 and LINE2. Using these font designations on language statements of Figure 6 would yield Figure 10.

Another datum which lends itself to add-on methodology is display color. The semantics might be millimicrons of wavelength (10^{-9} meters). This numeric representation is more precisely interpretable than subjective words (such as, red or green). The domain would represent the visible spectrum. The integers between 400 and 700 inclusive, INTG[400,700], could be used. Clearly, these add-on attributes may be attached to a given statement in arbitrary sequence. Figures 11 and 12 show the linear structure of Figure 2 with two add-ons.

For linear dimension, two entities, the nature of the dimension (e.g., vertical) and some intelligence on display of the text information, may be needed. This seems best accomplished by a new structure. For dimensioning and tolerancing semantics, ANSI6 should conform to the ANSI standard on dimensioning and tolerancing (ref. 2). For this simplified example, the language statements and tabular representations in Figure 13 may be used, with the structure constituents explained in Figure 14.

To represent fillet geometry implicitly, two entities, the fillet radius and some indication of the fillet direction, are required. As for linear dimension, a new structure type is indicated. The language statements and tabular representations of Figure 15 may be used.

To further demonstrate the use of new structures, consider the lattice work shown in Figure 16. This lattice may be considered as twelve bar elements totally fixed at the corners.

Figure 17 depicts the language statements and tabular representations for several of these corners. Also, the corners are shown as GRID points through association of a fixity flag to model degrees of freedom physically realizable at these junctions. Note that a BAR may be defined as a prism between two grid points (e.g., G2 and G1) with its cross-section area oriented in space through a third point (P3) per Figure 18. Definition of the BAR comprises sectional properties and material information attributes (see Figure 19 for a typical bar of the lattice). Finally, Figure 20 illustrates a way of capturing material properties by incorporating them into a new structure.

SUMMARY

Digital representation for communication of non-geometric product definition data is a vast subject. The ANS15 language constructs for basic shape data are extensible to other data categories. The two types of extensions presented herein with examples are adaptable to many aspects of product definition. Other extension concepts may be found in Appendix A of reference 1.

REFERENCES

1. Digital Representation for Communication of Product Definition Data. ANSI Y14.26M-1981, American Society of Mechanical Engineers, 1982.

2. Dimensioning and Tolerancing. ANSI Y14.5-1973, American Society of Mechanical Engineers, 1973.

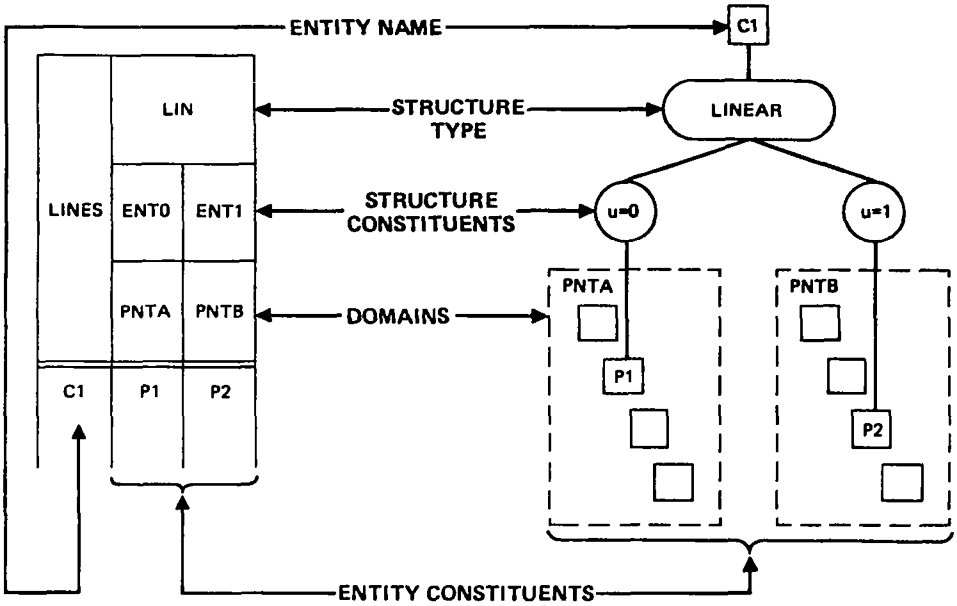
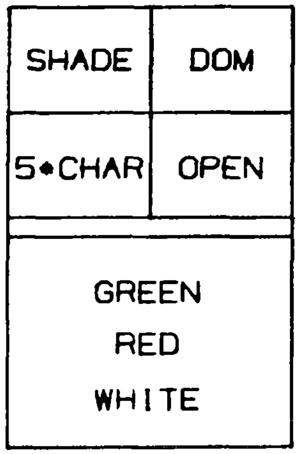


Figure 1.- ANSI5 structure and its parts.

LINEX	LIN	
	ENT0	ENT1
	POINT1	POINT1
L1	P1	P2
L2	P1	P4
L3	P3	P4

```
LINEX/LIN/ENT0,ENT1/2*POINT1:
 L1(P1,P2),L2(P1,P4),P3(P3,P4);
```

Figure 2.- ANSI5 linear structure.



SHADE/DOM/5*CHAR,OPEN: 'GREEN','RED','WHITE';

Figure 3.- ANSI5 domain structure.

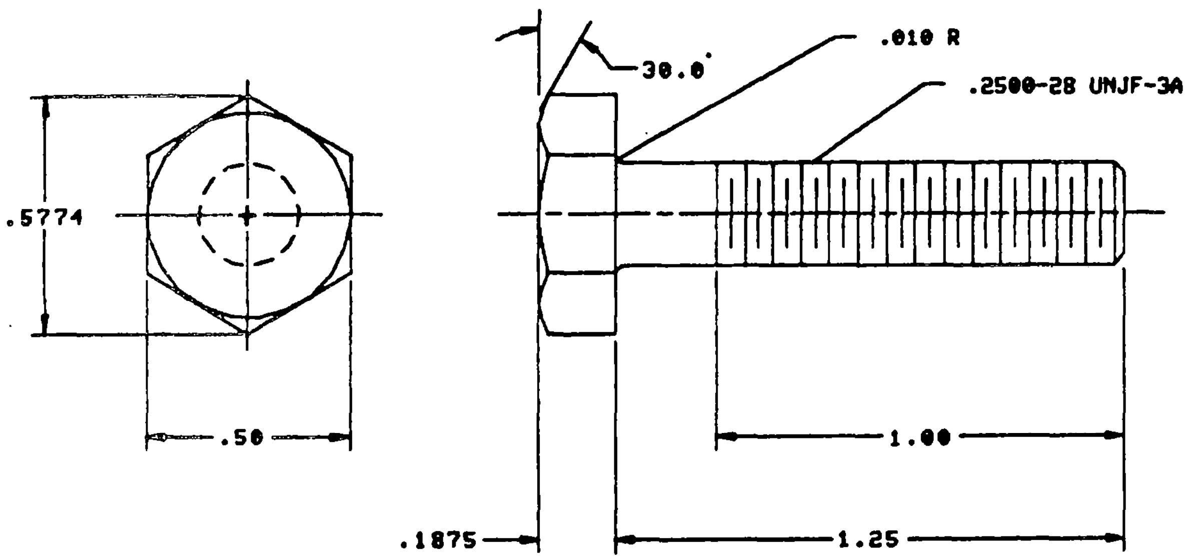


Figure 4.- Bolt drawing.

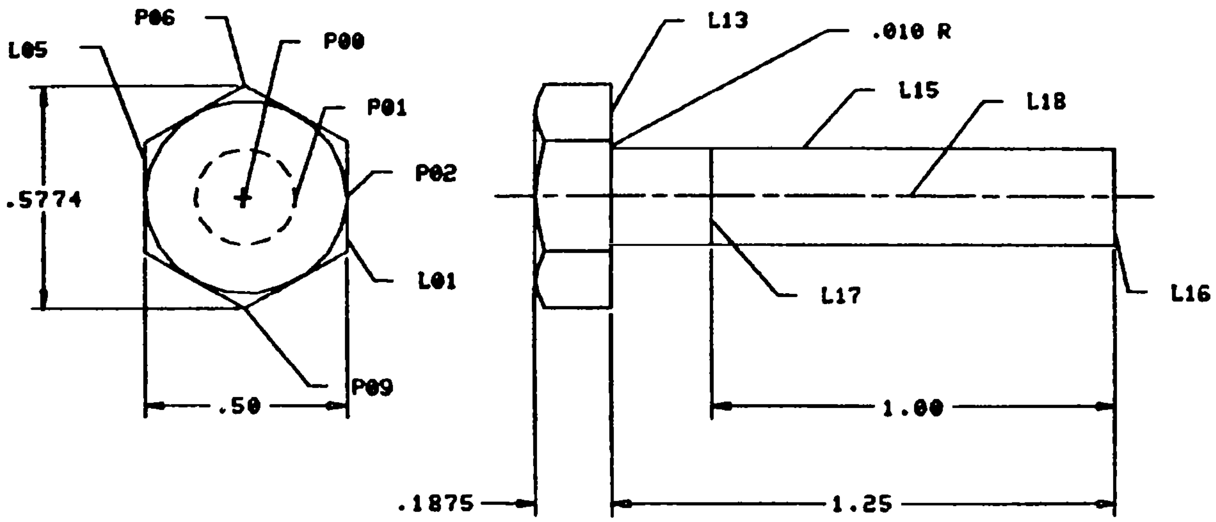


Figure 5.- Simplified bolt drawing.

```
POINT2/PNT/X,Y,Z/3*REAL:
    P00(-.75,0.0,0.0),P01(-.625,0.0.0.0),
    P02(-.5,0.0,0.0),P03(-.75,0.0,1.0),...;

LINE2/LIN/ENT0,ENT1/2*POINT2:
    L01(P04,P05),L02(P05,P06),...,L16(P17,P19),
    L17(P20,P21),L18(P12,P13);

CIRCS2/CIR/CTR,AXS,SPT/3*POINT2:
    CIR1(P00,P03,P01),CIR2(P00,P03,P02);

CONIC2/CON/ENT0,ENTA,ENT1,ENTB,ENT2,PAR1/5*POINT2,REAL(0.,1.):
    CON1(P26,P27,P10,P28,P29,.5),CON2(P29,P31,P32,P33,P34,.5),
    CON3(P34,P36,P11,P37,P38,.5);
```

Figure 6.- Bolt language statements.

POINT2	PNT		
	X	Y	Z
	REAL	REAL	REAL
P00	-.75	0.0	0.0
P01	-.625	0.0	0.0
P02	-.5	0.0	0.0
P03	-.75	0.0	1.0

LINE2	LIN	
	ENT0	ENT1
	POINT2	POINT2
L01	P04	P05
L02	P05	P06
L16	P17	P19
L17	P20	P21
L18	P12	P13

```
POINT2/PNT/X,Y,Z/3*REAL: P00(-.75,0.0,0.0),
P01(-.625,0.0,0.0),P02(-.5,0.0,0.0),P03(-.75,0.0,0.0),...;

LINE2/LIN/ENT0/ENT1/2*POINT2: L01(P04,P05),
L02(P05,P06),...,L16(P17,P19),L17(P20,P21),L18(P12,P13);
```

Figure 7.- Bolt: POINT2 and LINE2.

FONTC	DOM
7*CHAR	OPEN
SOLID	
DASH	
CENTER	
PHANTOM	
BLANK	

FONTP	DOM
6*CHAR	OPEN
POINT	
XHAIR	
BLANK	
CIRCLE	
SQUARE	

```
FONTC/DOM/7*CHAR,OPEN:
    'SOLID','DASH','CENTER','PHANTOM','BLANK';

FONTP/DOM/6*CHAR,OPEN:
    'POINT','XHAIR','BLANK','CIRCLE','SQUARE';
```

Figure 8.- Font domains.

POINT2	PNT			
	X	Y	Z	PFONT
	REAL	REAL	REAL	FONTP
P00	-.75	0.0	0.0	XHAIR
P01	-.625	0.0	0.0	NULL
P02	-.5	0.0	0.0	NULL
P03	-.75	0.0	1.0	NULL

LINE2	LIN		
	ENT0	ENT1	LFONT
	POINT2	POINT2	FONTC
L01	P04	P05	NULL
L02	P05	P06	NULL
L16	P17	P19	NULL
L17	P20	P21	NULL
L18	P12	P13	CENTER

```
POINT2/PNT/X,Y,Z,PFONT/3•REAL,FONTP: P00(-.75,0.0,0.0,'XHAIR'),
   P01(-.625,0.0,0.0,),P02(-.5,0.0,0.0,),P03(-.75,0.0,1.0,),...;

LINE2/LIN/ENT0,ENT1,LFONT/2•POINT2,FONTC: L01(P04,P05,),
   L02(P05,P06,),...,L16(P17,P19,),L17(P20,P21,),P18(P12,P13,'CENTER');
```

Figure 9.- Bolt: POINT2 and LINE2, augmented.

```
POINT2/PNT/X,Y,Z,PFONT/3•REAL,FONTP:
   P00(-.75,0.0,0.0,'XHAIR'),P01(-.625,0.0.0.0,),
   P02(-.5,0.0,0.0,),P03(-.75,0.0,1.0,),...;

LINE2/LIN/ENT0,ENT1,LFONT/2•POINT2,FONTC:
   L01(P04,P05,),L02(P05,P06,),...,L16(P17,P19,),
   L17(P20,P21,),L18(P12,P13,'CENTER');

CIRCS2/CIR/CTR,AXS,SPT,LFONT/3•POINT2,FONTC:
   CIR1(P00,P03,P01,'DASH'),CIR2(P00,P03,P02,);

CONIC2/CON/ENT0,ENTA,ENT1,ENTB,ENT2,PAR1/5•POINT2,REAL(0.,1.):
   CON1(P26,P27,P10,P28,P29,.5),CON2(P29,P31,P32,P33,P34,.5),
   CON3(P34,P36,P11,P37,P38,.5);
```

Figure 10.- Augmented bolt language.

LINEX	LIN	
	ENTO	ENT1
	POINT1	POINT1
L1	P1	P2
L2	P1	P4
L3	P3	P4

LFONT
FONTC
DASH
NULL
CENTER

COLOR
INTG [400,700]
475
540
670

Figure 11.- Two add-on attributes.

LINEX	LIN			
	ENTO	ENT1	LFONT	COLOR
	POINT1	POINT1	FONTC	INTG [400,700]
L1	P1	P2	DASH	475
L2	P1	P4	NULL	540
L3	P3	P4	CENTER	670

```
LINEX/LIN/ENTO,ENT1,LFONT,COLOR/2*POINT1,FONTC,INTG[400,700]:
   L1(P1,P2,'DASH',475),L2(P1,P4,,540),L3(P3,P4,'CENTER',670);
```

Figure 12.- Linear structure with two add-ons.

LDIMTP	DOM
7*CHAR	OPEN
VERT HORIZ TRUE	

DIMI	LDIM					
	ENT1	ENT2	TYPE	PREC	BPT	ANGLE
	POINT2 .OR. LINE2	POINT2 .OR. LINE2	LDIMTP	INTG [0,7]	POINT2	REAL
LD1	L05	L01	HORIZ	2	P51	0.0
LD2	P06	P09	VERT	4	P52	0.0

```
LDIMTP/DOM/5*CHAR,OPEN:'VERT','HORIZ','TRUE';

DIMI/LDIM/ENT1,ENT2,TYPE,PREC,BPT,ANGLE/
      2*POINT2.OR.LINE2,LDIMTP,INTG,POINT2,REAL:
   LD1(L05,L01,'HORIZ',2,P51,0.),LD2(P06,P09,'VERT',4,P52,0.),...;
```

Figure 13.- Linear dimension.

ENT1, ENT2	**—**	**REFERENCE ENTITIES**
TYPE	**—**	**TYPE OF LINEAR DIMENSION**
PREC	**—**	**PRECISION: NUMBER OF DECIMAL DIGITS**
BPT	**—**	**BASE POINT FOR LOCATION OF DERIVED TEXT**
ANGLE	**—**	**ANGLE OF DERIVED TEXT**

Figure 14.- Linear dimension structure constituents.

FILLET	FLET			
	ENT1	ENT2	RAD	NPT
	LINE2	LINE2	REAL	POINT2
FL1	L13	L15	0.010	P57

FILLET/FLET/ENT1,ENT2,RAD,NPT/2•LINE2,REAL,POINT2:
FL1(L13,L15,0.010,P57);

Figure 15.- Implied fillet geometry.

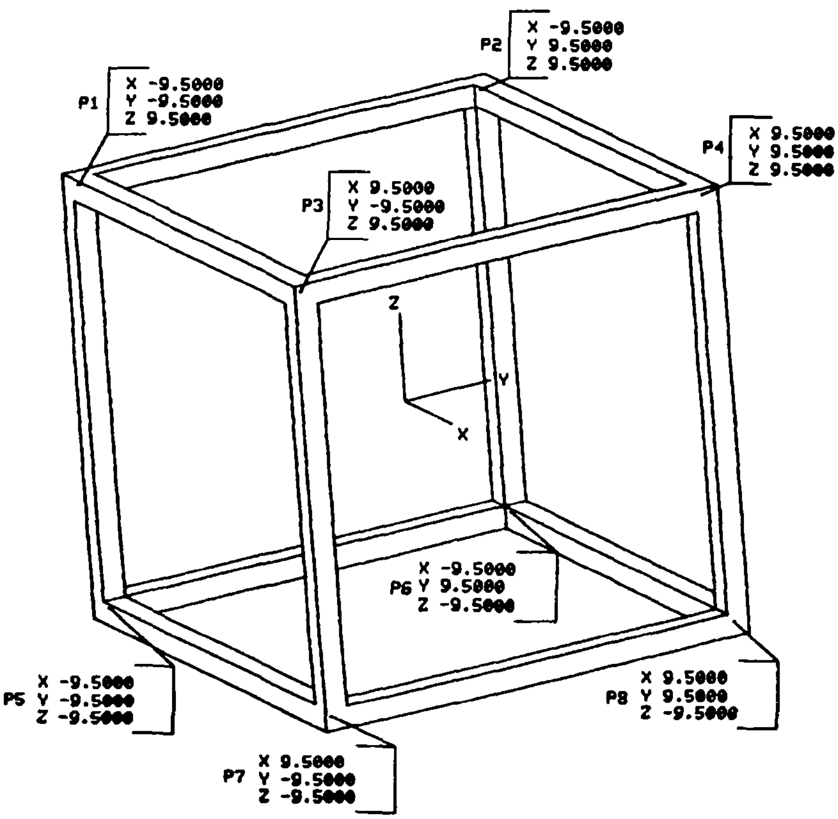


Figure 16.- Lattice work.

POINT3	PNT		
	X	Y	Z
	REAL	REAL	REAL
P1	-9.500	-9.500	9.500
P2	-9.500	9.500	9.500
P3	9.500	-9.500	9.500

GPNT	GRID	
	LOC	FIX
	POINT3	FFLAG
G1	P1	0
G2	P2	0
G3	P3	0

```
POINT3/PNT/X,Y,Z/3*REAL:
   P1(-9.500,-9.500,9.500),P2(-9.500,9.500,9.500),P3(9.500,-9.500,9.500),...:
GPNT/GRID/LOC,FIX/POINT3,FFLAG: G1(P1,0),G2(P2,0),G3(P3,0),...:
```

Figure 17.- Lattice: POINT3 and GPNT.

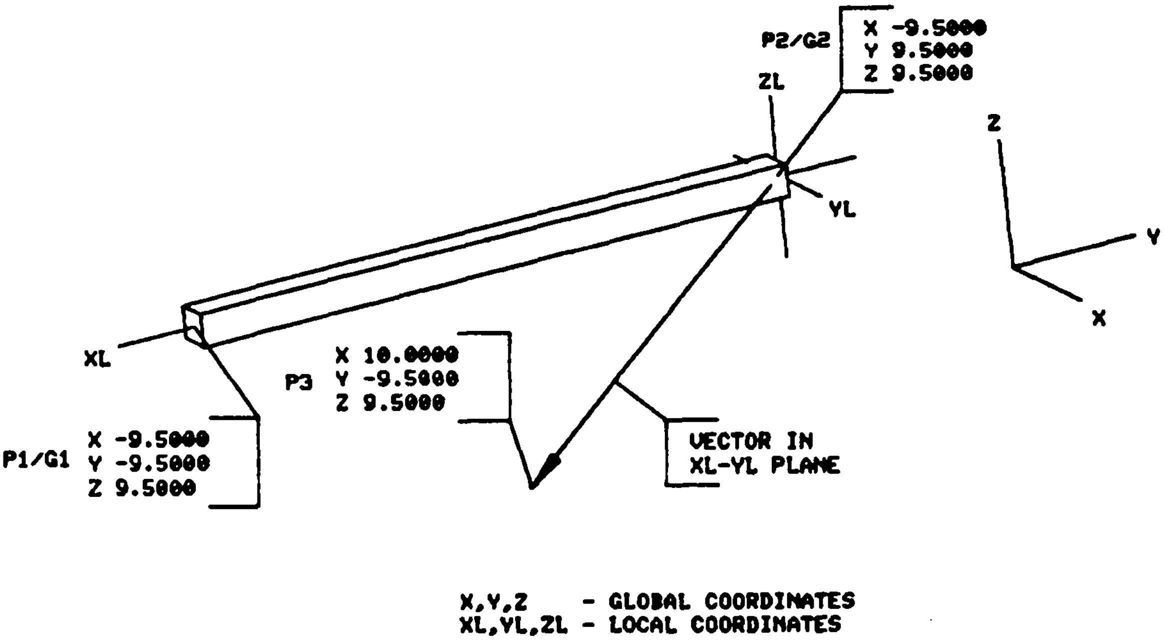


Figure 18.- Representative bar element.

BAR1	BAR								
	END1	END2	ORIENT	AREA	IYY	IZZ	IYZ	J	MAT
	GPNT	GPNT	POINT3	REAL	REAL	REAL	REAL	REAL	MATL
B12	G2	G1	P3	1.0000	0.0833	0.0833	0.0000	0.1667	AL

```
BAR1/BAR/END1,END2,ORIENT,AREA,IYY,IZZ,IYZ,J,MAT/2•GPNT,POINT3,5•REAL,MATL:
   B12(G2,G1,P3,1.0000,0.0833,0.0833,0.0000,0.1667,AL),...;
```

Figure 19.- Lattice: BAR1.

MATL	MAT			
	E	G	MU	RHO
	REAL	REAL	REAL	REAL
AL	10.5E6	4.0E6	0.33	0.098
STEEL	30.0E6	12.0E6	0.288	0.283

E: YOUNG'S MODULUS (PSI)

G: SHEAR MODULUS (PSI)

MU: POISSON RATIO

RHO: DENSITY (LB/CU.IN.)

```
MATL/MAT/E,G,MU,RHO/4•REAL:
   AL(10.5E6,4.0E6,0.33,0.098),STEEL(30.0E6,12.0E6,0.288,0.283),...;
```

Figure 20.- Lattice: MATL.

MODELING CONCEPTS FOR COMMUNICATION OF GEOMETRIC SHAPE DATA

M. F. Collins
Control Data Corporation
Arden Hills, Minnesota

R. F. Emnett
McDonnell Douglas Automation Company
St. Louis, Missouri

R. L. Magedson
Boeing Commercial Airplane Company
Seattle, Washington

H. H. Shu
McDonnell Douglas Astronautics Company
St. Louis, Missouri

INTRODUCTION

ANSI5, an abbreviation for Section 5 of the American National Standard under Engineering Drawing and Related Documentation Practices (Committee Y14) on Digital Representation for Communication of Product Definition Data (ANSI Y14.26M-1981), allows encoding of a broad range of geometric shapes to be communicated through digital channels (ref. 1). This paper is a brief review of its underlying concepts.

The intent of ANSI5 is to devise a unified set of concise language formats for transmission of data pertaining to five types of geometric entities in Euclidean 3-space (E^3). These may be regarded as point-like, curve-like, surface-like, solid-like, and a combination of these types. For the first four types, ANSI5 makes a distinction between the geometry and topology. Geometry is a description of the spatial occupancy of the entity, and topology discusses the interconnectedness of the entity's boundary components.

GEOMETRY

Geometrically, ANSI5 entities are point sets in E^3 subject to rigorously defined set membership rules known as geometric structures.

The first kind of geometric structure defines the (X,Y,Z) coordinates of each point in the set using parametric representation. In general, a point has a parametric dimensionality zero; a curve, one; a surface, two; and a solid, three. ANSI5 has fifteen geometric structures of the parametric kind in the following four categories:

- o Primitive
  - Point
  - Circle
- o Interpolative (Blending)
  - Linear
  - Circular Arc
  - Conic Arc
  - Cubic
  - N-th Degree Polynomial
  - Spline
- o Generative (Sweeping)
  - Translation/Rotation
  - Rotation
  - Translation
- o Special
  - Reverse Curve
  - Curve String
  - Evaluation (Restriction)
  - Flip (Surface Normals)

The second kind of geometric structure addresses the Boolean combinations of pre-defined entities (point sets) of homogeneous or mixed dimensionality. There are four structures under the category of point set.

- Dimension-Selecting Closure
- Union
- Intersection
- Difference

The third kind of geometric structure is replicative. Five methods for entity replications are allowed under ANSI5.

- Translation/Rotation
- Rotation
- Translation
- Scale
- Reflection

CONNECTIVITY AND GROUPING

Topologically, the boundaries of ANSI5 objects can be expressed in logical combinations of the following topological entities:

- Vertex
- Edge
- Loop
- Face
- Shell
- Object

For modeling geometry, ANSI5 provides a set of seven (the six above plus SWITCH for reversal of edge direction) well-formulated definitions for these concepts, known as topological structures. Each VERTEX, EDGE, FACE, and OBJECT structure also provides a linkage to a corresponding geometric entity for its spatial location in E^3. Refer to figure 1 for geometry/topology relationships.

Finally, any collection of geometric and/or topological entities may conveniently be referenced as one entity under the GROUP structure. One use of this structure is to represent assemblies of objects.

Appendix A of Y14.26M-1981 (ref. 1) contains suggestions and examples for possible extensions into other areas not currently addressed by ANSI5. These are intrinsic and derivable attributes of an entity, alternative geometric descriptions, and information other than shape data.

LANGUAGE ELEMENTS

The language elements for expressing the geometric shape data just outlined are entities, structures, relations, and domains.

The entities are the points, curves, surfaces, solids, and mixed-dimension data being communicated by the language, and each is an instance of one of the 32 structures. Most of the geometric structures are efficiently designed to be independent of dimension. For example, the linear interpolation structure (named LIN) is used for a line between two points, a ruled surface between two curves, and also a "ruled solid" between two surfaces.

A named formal collection of named entities of the same structure type is the basic statement or "sentence" of the language. It is called a relation because of its similarity to the mathematical concept of the same name. A mathematical relation is a collection of ordered lists of the same length whose entries come from an ordered list of sets, called domains. The term "domain" for the fourth ANSI5 language element serves a completely analogous function.

The specification for the general linear interpolation structure language statement is shown in figure 2, with corresponding components arranged in tabular form to emphasize component relationships. Each one of the right two columns is headed by a LIN formal structure constituent, which is directly above the corresponding domain set name, which in turn is directly above the names of entities taken from that domain. The relation name is at the far left, and the defined entity names are the bracketed IDn terms. In practice, in digital data communication, such a helpful layout is not needed, and a much more compact (however, still human-readable) style would be used, as in the following actual language statement.

```
LNEXAM/LIN/ENT0,ENT1/2*PT1.OR.CV1.OR.SF1:
  L1(P1,P2),SR1(C1,C2),VOL(BOTTOM,TOP);
```

DIMENSION-SELECTING CLOSURE

The dimension-selecting closure (DSEL) structure in ANSI5 is of particular interest for its apparent uniqueness in the literature, its function in the ANSI5 language in conjunction with the other point set structures, and its relationship to other concepts in mathematics (e.g., regularized Boolean operations). Figures 5-26 to 5-28 of ANSI5 (ref. 1) are reproduced here as figures 3 to 5 to illustrate this structure.

EXTENDED EXAMPLE

A typical example is presented in table 1 (language) and in figures 6 to 22 (graphics). It has been constituted to bring out several important aspects of ANSI5 use.

SUMMARY

In summary, ANSI5 provides a set of thirty-two structures and their language formats for the description and communication of basic shapes in 3-dimensional space. Its geometric coverage is sufficiently broad to cover shape data employed not only in most CAD/CAM systems commercially available today, but also in those under active consideration by vendors for the near future. It is based on consistent and rigorous mathematical formulation and will be a stable foundation upon which other aspects of product definition data may rest.

REFERENCE

1. American National Standards Institute: Digital Representation for Communication of Product Definition Data, American National Standard ANSI Y14.26M-1981. American Society of Mechanical Engineers, New York, 1982.

TABLE 1. - LANGUAGE OF EXTENDED EXAMPLE

```
POLYPNTS 'POINTS FOR CURVE DEFINITION + ROTATION'
  /PNT/X,Y,Z/3*REAL:PP0(0.,0.,0.),PP1(1.,.3,0.),
  PP2(2.,1.6,0.),PP3(3.,2.7,0.),PP4(4.,0.,0.),
  PAX(4.,0.,1.),PNEW(8.,0.,0.);
CURV1 'QUARTIC POLYNOMIAL CURVE'/POLY/N,(N+1)*ENT,
  (N-1)*PAR/INTG,(N+1)*POLYPNTS,(N-1)*REAL(0.,1.):
  HILL(4,PP0.PP1,PP2,PP3,PP4,.25,.50,.75);
CURV2 'ROTATION OF HILL IN  X-Y  PLANE ABOUT PP4'
  /RRP/ENT,Q1,Q2,Q3,R3/CURV1,4*POLYPNTS:
  TURN(HILL,PP4,PAX,PP0,PNEW);
CURV3 'REVERSAL OF PARAMETER DIRECTION FOR STRING'
  /REV/CRV/CURV2:DALE(TURN);
BASEPNTS 'POINTS FOR FINISHING BASE DEFINITION'
  /PNT/X,Y,Z/3*REAL:PURNORM(9.5,0.,11.),
  PUL(-1.5, 0.0,0.),PUR(9.5. 0.0,0.),
  PLL(-1.5,-3.5,0.),PLR(9.5,-3.5,0.);
BASELINS 'LINES FOR FINISHING BASE DEFINITION'
  /LIN/ENT0,ENT1/2*BASEPNTS.OR.POLYPNTS:
  FLAT1(PUL,PP0).FLAT2(PNEW,PUR).BOTTOM(PLL,PLR),
  BASEVEC(PUR,PURNORM);
BLOKPNTS 'POINTS FOR TOP BLOCK'/PNT/X,Y,Z/3*REAL:
  KLL(-3.,0.,3.),KLR(8.,0.,3.),KUL(-3.,3.5,3.),
  KUR(8.,3.5.3.),KURNORM(8.,3.5,7.);
BLOKLINS 'LINES FOR TOP BLOCK'/LIN/ENT0,ENT1/
  2*BLOKPNTS:L1(KUL,KUR),L2(KLL,KLR),
  L3(KUR.KURNORM);
COMP 'COMPOSITE CURVE'/CSTR/N,N*CRV.(N-1)*PAR/
  INTG,N*BASELINS.OR.CURV1.OR.CURV3,(N-1)*REAL:
  BASETOP(4,FLAT1,HILL,DALE,FLAT2,.136,.5,.864);
BACK 'SOLID BACK SURFACES'/LIN/ENT0,ENT1/
  2*COMP.OR.BASELINS.OR.BLOKLINS:
  BACKBLOK(L1,L2).BACKBASE(BASETOP,BOTTOM);
SWEEP 'STRAIGHT LINE SWEPT SOLIDS'/TGN/ENT,TRL/
  BACK.BLOKLINS.OR.BASELINS:
  BASE(BACKBASE,BASEVEC),BLOK(BACKBLOK.L3);
COMMON 'BOOLEAN INTERSECTION OF SOLIDS'/INT/
  N.N*ENT/INTG,N*SWEEP:BOTH(2,BASE.BLOK);
PURE 'ISOLATE DIMENSIONALLY HOMOGENEOUS SUBSETS'
  /DSEL/DIM,ENT,NPT/INTG.COMMON,BASEPNTS:
  CL3(3,BOTH,NULL) 'MAXIMAL SOLID SUBSET OF BOTH',
  CL2(2,BOTH,NULL) 'ISOLATED SURFACE(S) OF BOTH',
  CL1(1,BOTH,NULL) 'ISOLATED CURVE(S) OF BOTH',
  CL0(0,BOTH,NULL) 'WHATEVER REMAINS OF BOTH',
  CLII(2,BOTH,PUR) 'DE FACTO EQUALITY WITH CL2';
```

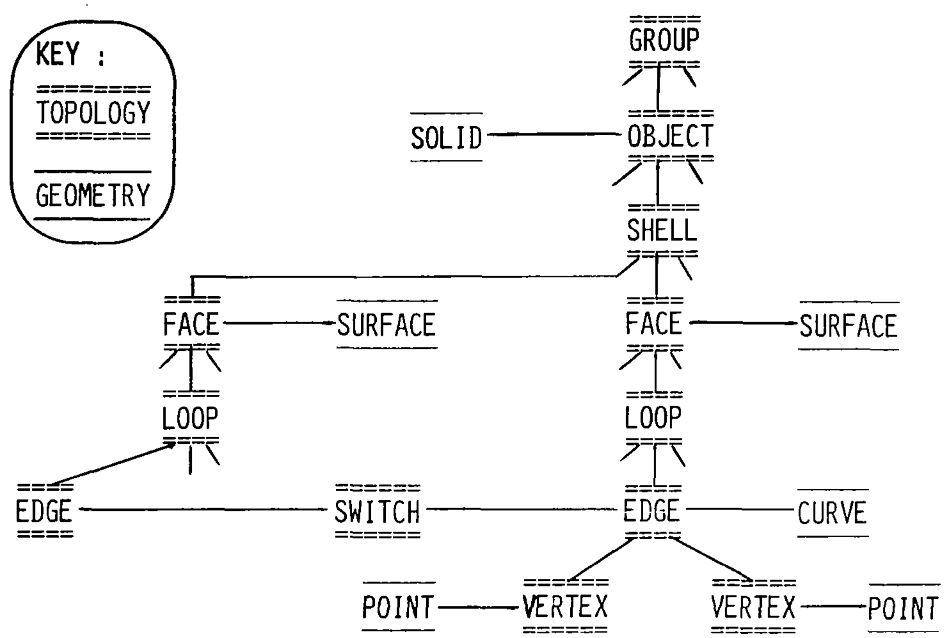


Figure 1.- Geometry/topology relationships.

$$
\begin{array}{llll}
[\mathrm{REL.NAME}]\ /\ \mathrm{LIN}\ /\ & \mathrm{ENT0} & , \ \mathrm{ENT1} & / \\
 & [\mathrm{DOM1}] & , \ [\mathrm{DOM2}] & : \\
[\mathrm{ID1}](& [\mathrm{ENT}_{1,0}] & , \ [\mathrm{ENT}_{1,1}]\ ) & , \\
[\mathrm{ID2}](& [\mathrm{ENT}_{2,0}] & , \ [\mathrm{ENT}_{2,1}]\ ) & , \\
[\mathrm{ID3}](& [\mathrm{ENT}_{3,0}] & , \ [\mathrm{ENT}_{3,1}]\ ) & , \\
 & \vdots & & \\
[\mathrm{IDi}](& [\mathrm{ENT}_{i,0}] & , \ [\mathrm{ENT}_{i,1}]\ ) & ;
\end{array}
$$

Figure 2.- LINear structure language statement.

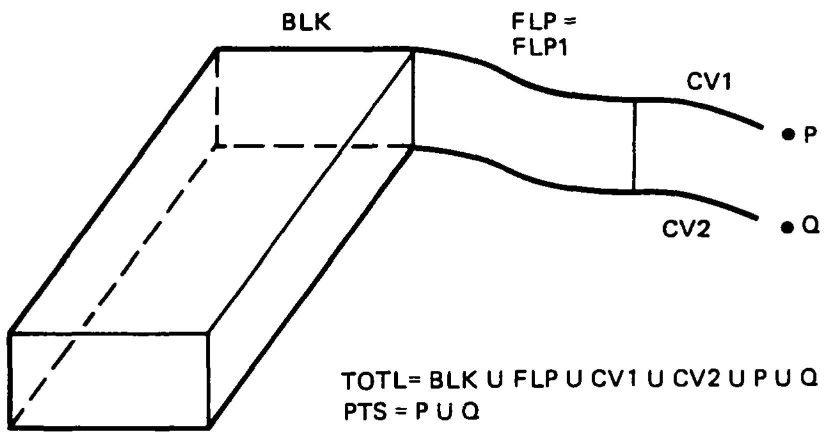


Figure 3.- Sets resulting from dimension-selecting closures of TOTL. (From ref. 1.)

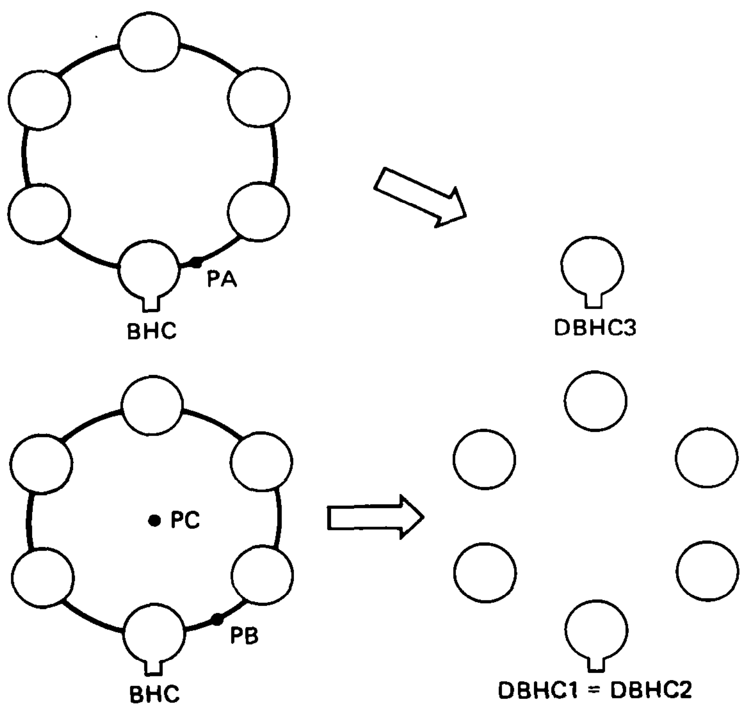


Figure 4.- Sets resulting from dimension-selecting closures of BHC. (From ref. 1.)

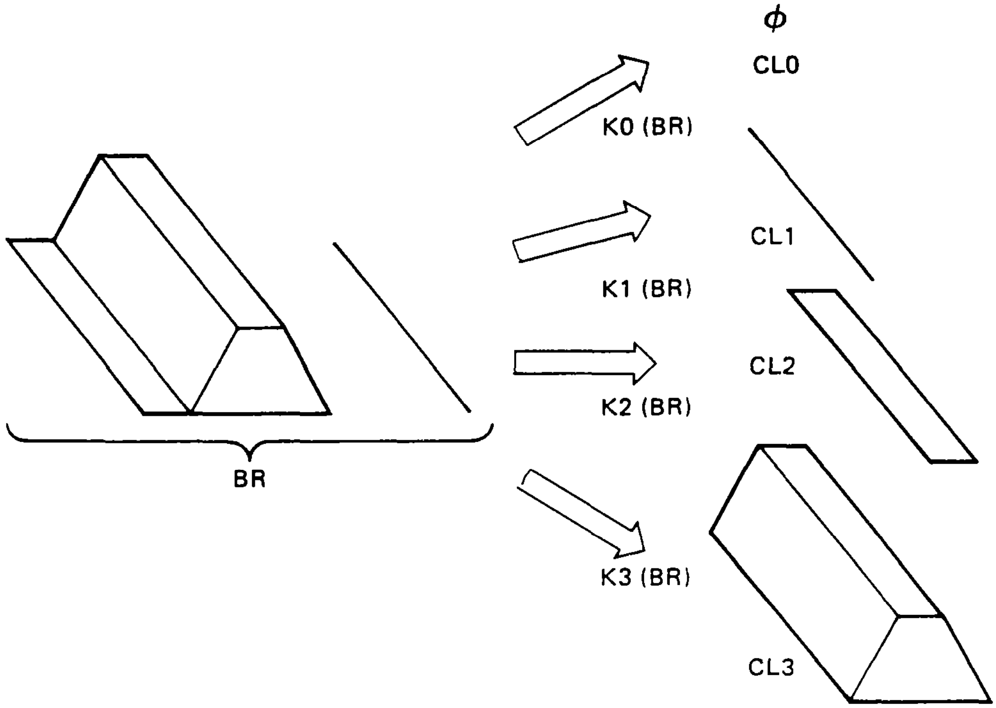


Figure 5.- Sets resulting from dimension-selecting closures of BR. (From ref. 1.)

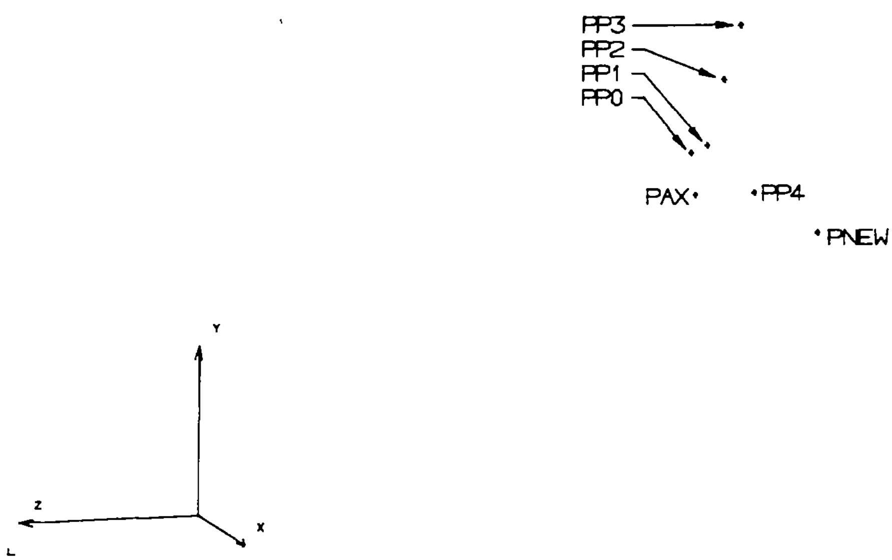


Figure 6.- Points for curve definition + rotation.

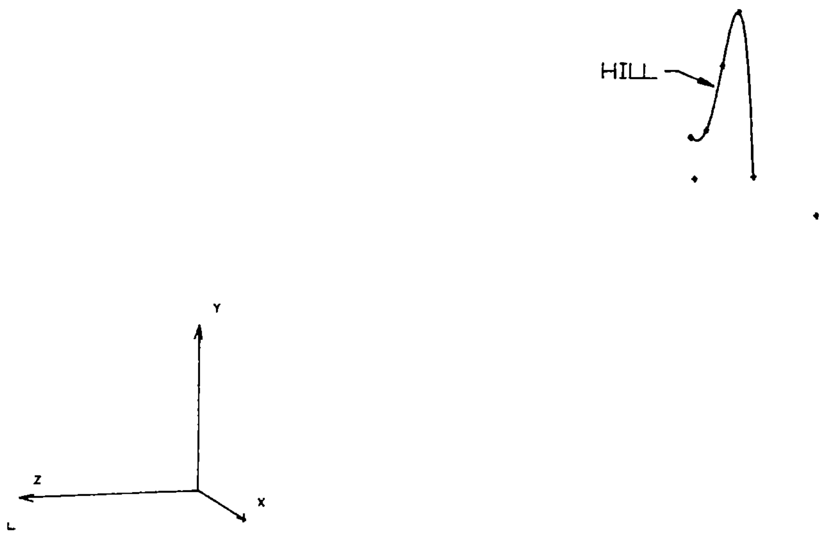


Figure 7.- Quartic polynomial curve.

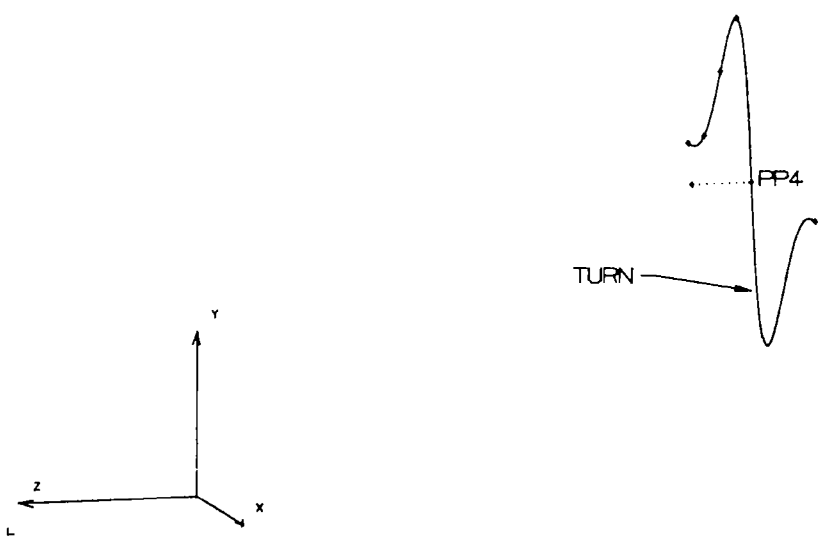


Figure 8.- Rotation of HILL in X-Y plane about PP4.

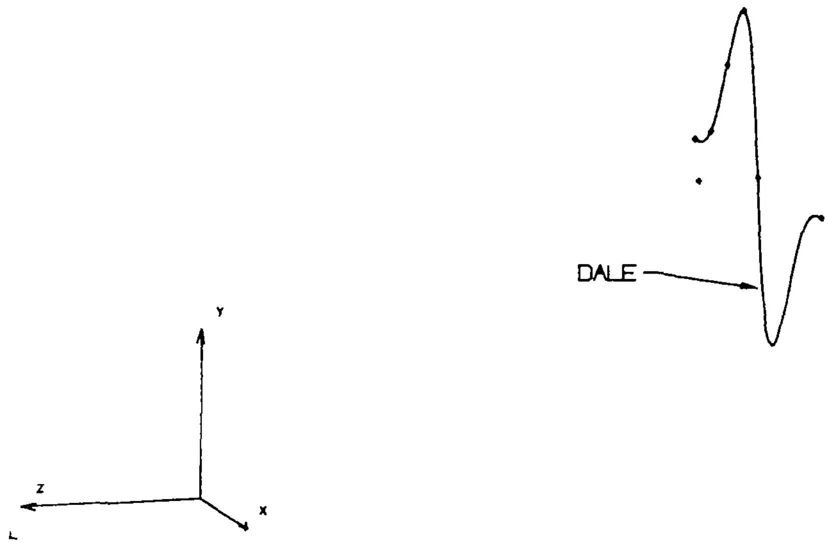


Figure 9.- Reversal of parameter direction for string.

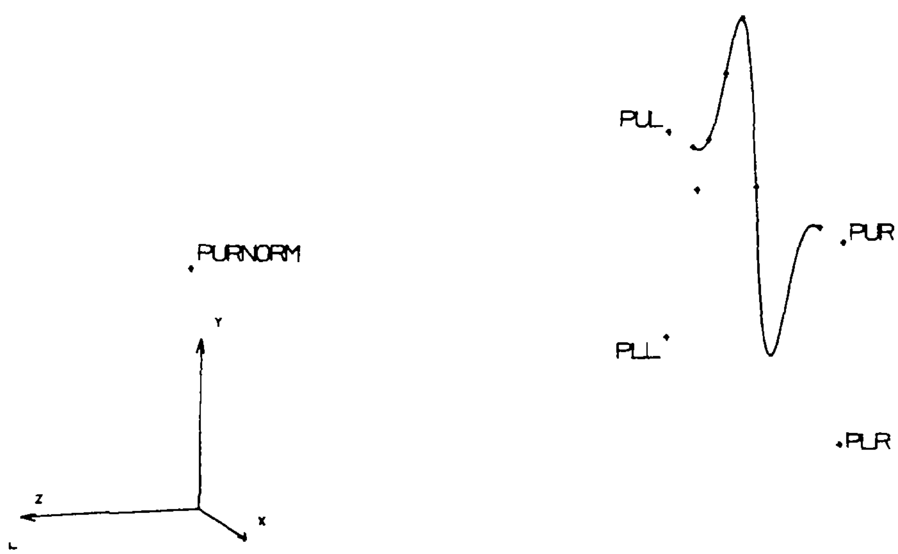


Figure 10.- Points for finishing base definition.

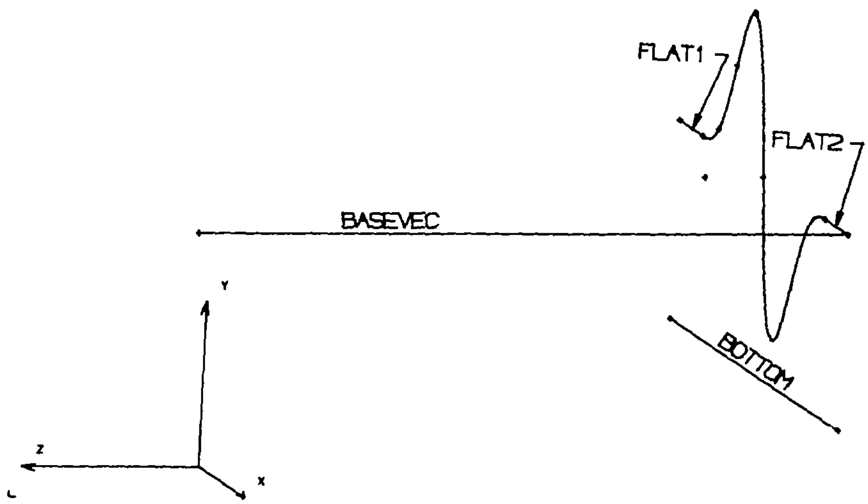


Figure 11.- Lines for finishing base definition.

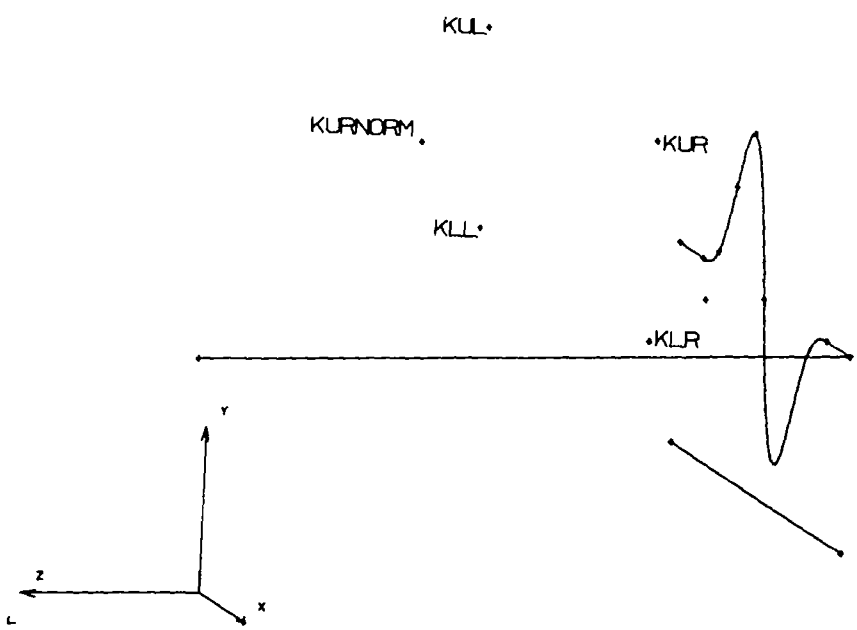


Figure 12.- Points for top block.

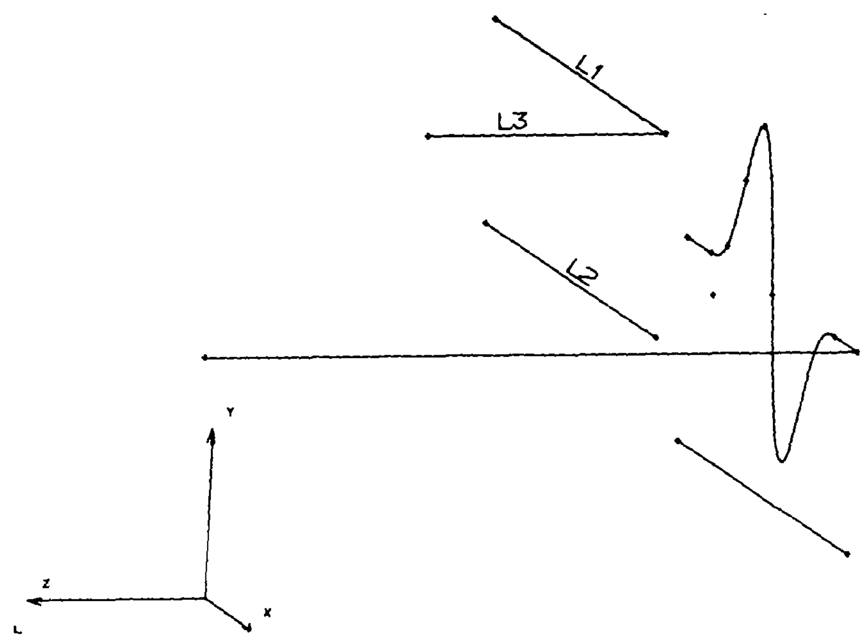


Figure 13.- Lines for top block.

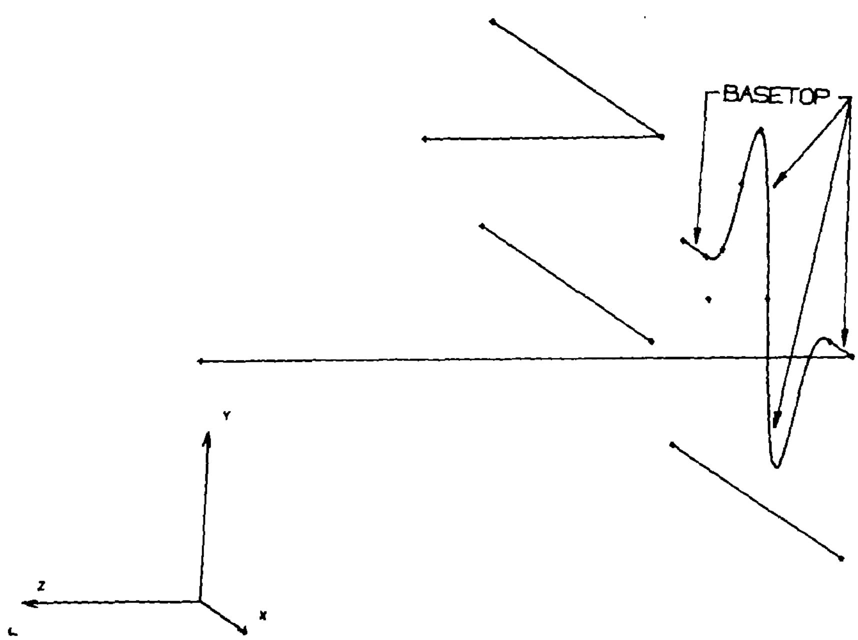


Figure 14.- Composite curve.

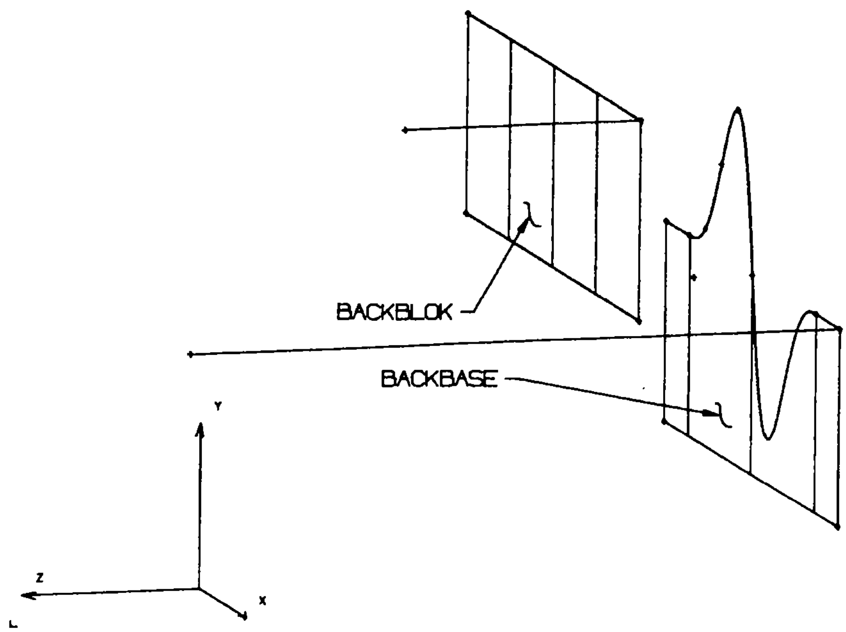


Figure 15.- Solid back surfaces.

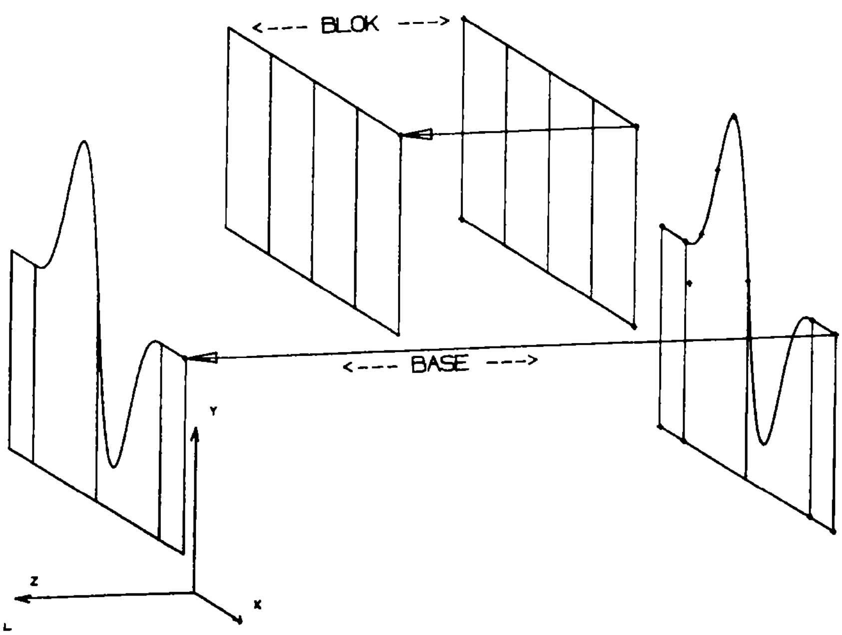


Figure 16.- Straight line swept solids.

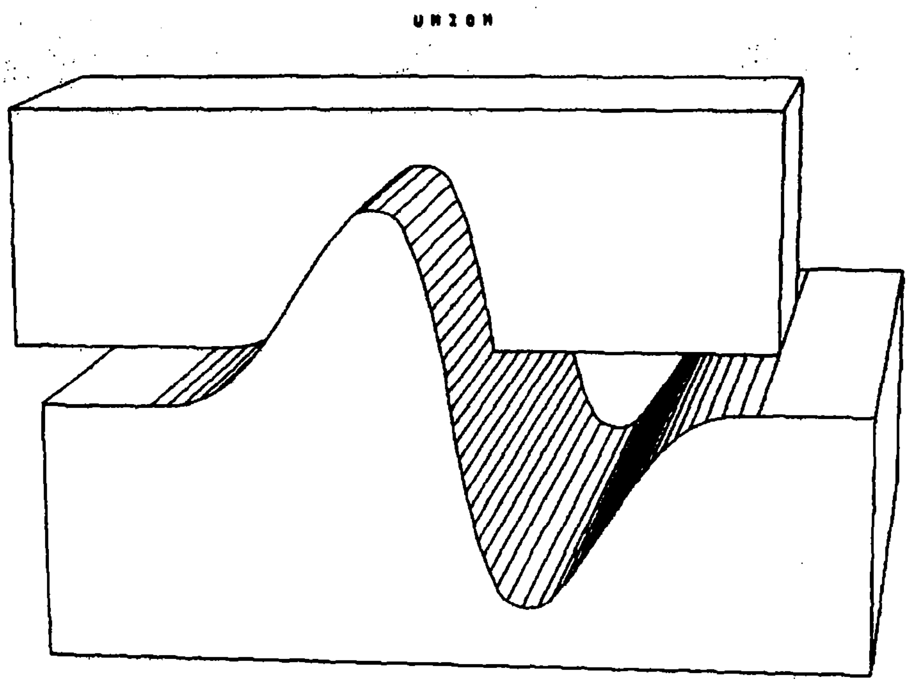


Figure 17.- BASE and BLOK, original positions.

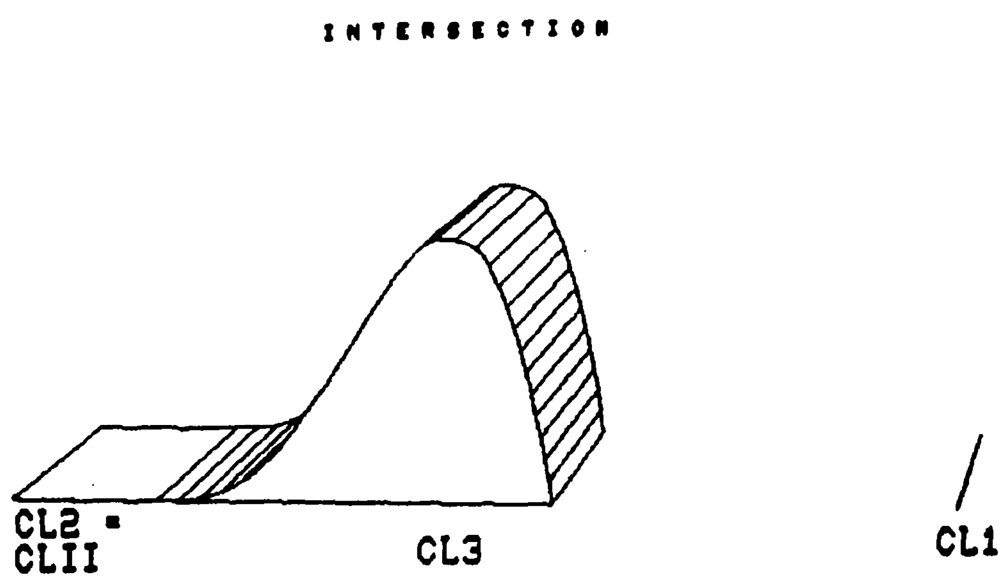


Figure 18.- DSEL results (fig. 17).

INTERSECTION

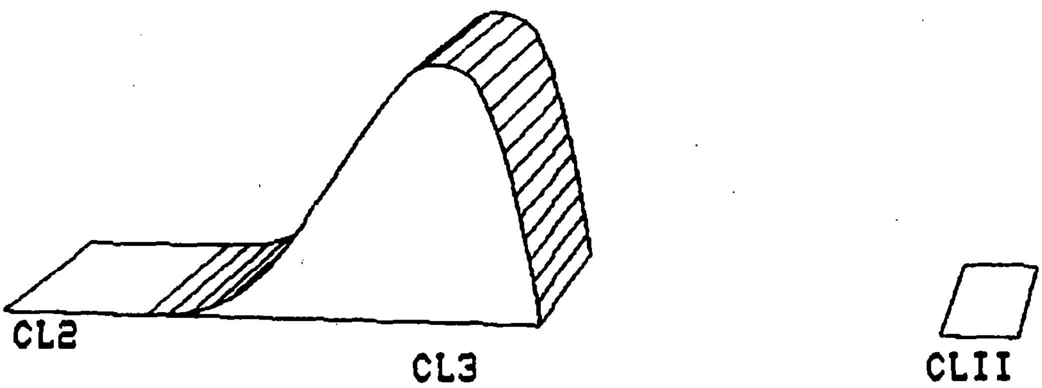


Figure 19.- Slide BLOK to right along BASE.

UNION

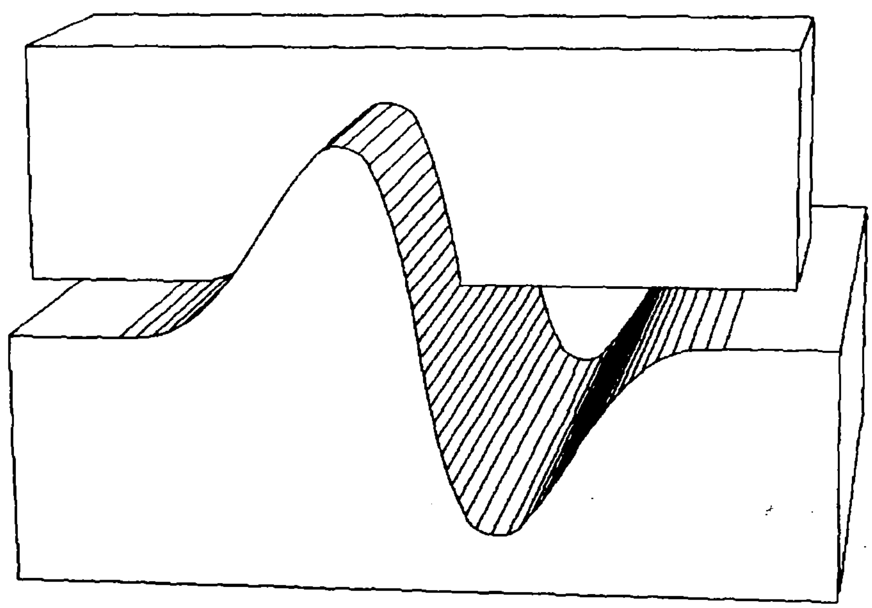


Figure 20.- DSEL results (fig. 19).

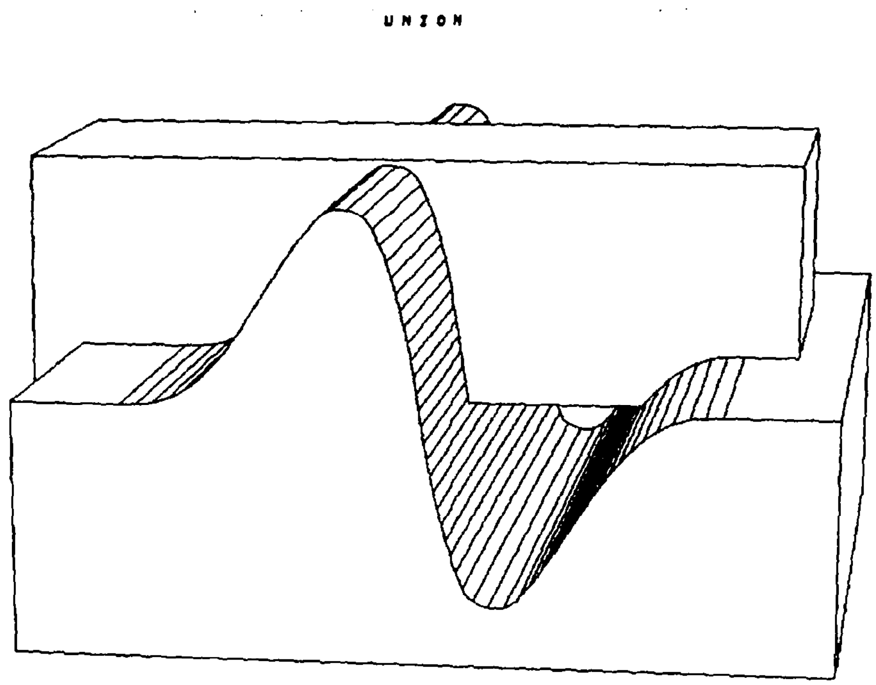


Figure 21.- Lower BLOK into BASE.

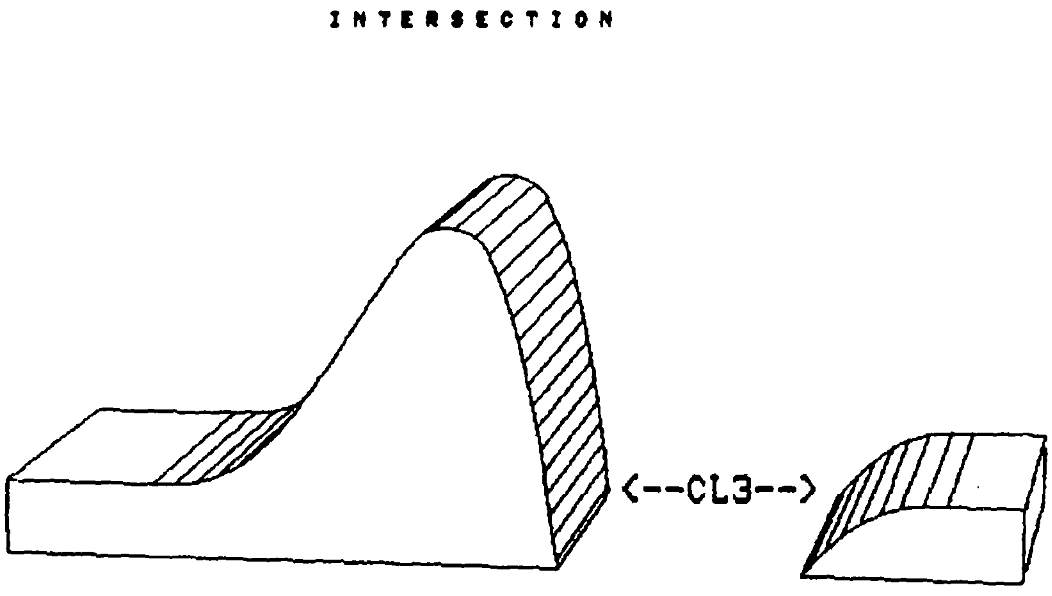


Figure 22.- DSEL results (fig. 21).

COMPUTER-AIDED SURFACE REPRESENTATION AND DESIGN

Robert E. Barnhill
University of Utah
Salt Lake City, Utah

This research creates new methods for representing and approximating three- and four-dimensional surfaces. These schemes are the core of our forthcoming Surface Software Package. Surfaces can be understood adequately only through the use of dynamic computer graphics; therefore, graphics capability is also included. The numerous applications of surface methods include modeling physical phenomena (e.g., combustion) and designing objects (e.g., airplanes and cars).

In addition to three-dimensional surfaces, there are interesting four-dimensional "surfaces", such as temperature as a function of the three spatial variables. Because the geometric information for these problems can be located arbitrarily in three- or four-dimensional space, the surface schemes must be able to handle arbitrarily located data. The standard (and easier) approach to surfaces, which uses tensor products of curves, restricts the surface method's applicability to (rectangularly) "gridded" data. We take the more ambitious approach of devising robust surface methods that are applicable to arbitrarily located data. There are two broad classes of methods suitable for solving these problems (i.e., problems for which simplifying geometric assumptions cannot be made):

1. Surface interpolants defined over triangles or tetrahedra

2. Surface interpolants defined directly from the given data

Users ordinarily want smoother surfaces than their data imply directly, so that additional information sometimes must be created. (A notable feature of our methods is that the smoothness of the surface is always greater than or equal to the smoothness of the data. This is not true of most other schemes, such as finite-element schemes.) Rendering three- and four-dimensional surfaces in a form maximally useful to the user is another interesting and difficult problem. We are using interactive computer graphics to display three-dimensional surfaces and will add color to help visualize and interpret four-dimensional surfaces.

PRESENT CAPABILITIES AND FUTURE REQUIREMENTS FOR COMPUTER-AIDED GEOMETRIC MODELING IN THE DESIGN AND MANUFACTURE OF GAS TURBINES

E. Caille, M. Propen, and A. Hoffman
Avco Lycoming Division
Stratford, Connecticut

INTRODUCTION

Gas turbine engine design requires the ability to rapidly develop complex structures which are subject to severe thermal and mechanical operating loads. As in all facets of the aerospace industry, engine designs are constantly driving towards increased performance, higher temperatures, higher speeds, and lower weight. The ability to address such requirements in a relatively short time frame has resulted in a major thrust towards integrated design/analysis/manufacturing systems. These computer-driven graphics systems represent a unique challenge, with major payback opportunities if properly conceived, implemented, and applied.

Detailed geometric models can form the basis for subsequent iterative optimization relying on the association between the geometry, finite element models, and manufacturing and inspection parameters. The major problems with the development of such a generalized system have been the lack of three-dimensional architecture for arbitrary geometric entities and the lack of a comprehensive distributed database. To be an effective design tool, all geometric changes must be reflected in analysis models, N/C files, etc. This approach has been implemented successfully for specialized component design such as airfoils and impellers, which are critical gas turbine elements.

Increasing dependence on interactive design systems has also brought out the need for reliable distributed systems with minimal impact on individual users. The emphasis has therefore shifted to a distributed environment with database compatibility requirements between the micro, the mini, and the mainframe computers. Relational databases with multiple levels of accessibility to data subsets appear to be promising for satisfying this requirement.

The effective utilization of interdisciplinary graphics systems also requires an acute awareness of the strengths and limitations of available hardware components. The cost-effective system requires a fine balance of central and distributed processors and intelligent terminals. From the user's perspective, judicious use of storage, raster, and vector display technology can provide productive results.

IMPELLER DESIGN PROCESS

The efficient transfer of information between disciplines require development of component-specific programs. These programs address the unique functional requirements of the individual tasks while maintaining a structured flow of information throughout the total system. The implementation of an interactive-impeller CAD/CAM system was achieved through the integration of specialized in-house developed software and commercially available packages.

The impeller design system consists of VAX 11/780 based interactive graphics programs in which aerodynamic analysis, detailed geometric modeling, finite-element pre-processing, five-axis N/C tape generation, and automated quality inspection data files are prepared.

In the aerodynamic analysis program, the designer interactively examines the resulting blade shape based on pre-specified flow and load conditions. Once the geometry has been generated, aerodynamic iteration continues until acceptable surface velocities and efficienci are attained.

Structural and dynamic analyses are performed using NASTRAN. The association of the geometry with internal finite-element parameter enables the designer to construct a new model for any design change. This process continues until all structural objectives are achieved.

The manufacturing of the impeller vane surface is accomplished using a five-axis N/C milling machine. A currently available software package which uses geometric design data to produce five-axis N/C milling machine tapes has been interfaced with the impeller design system.

This comprehensive design system can take the development of a critical gas turbine component, such as an impeller, from the conceptual stage through manufacturing in a matter of weeks. This concept is illustrated in Figure 1.

GENERATING 3-D GEOMETRY

The components of turbine engines are varied and complex in their geometric definition, as illustrated by the turbine nozzle model in Figure 2. If the design of these parts is to proceed in a direction other than just creating 2-D layouts, the structure of graphics systems must change dramatically. The general approach of creating 3-D geometry interactively presents three problems:

1. The available building blocks are usually inadequate.

2. The application of the building blocks is difficult.

3. Visualization of a 3-D part is difficult.

The current inventory of building blocks provided by CAD/CAM vendors is comprised of points, lines, conics, and splines. For creating 2-D geometric definitions this is more than adequate. However, when these building blocks are applied to a 3-D problem, they fall short in their usefulness. These entities cannot completely define an arbitrary 3-D engine component. They are capable of defining the edges but never the faces of a component.

The use of surfaces to define the faces of a component has some potential. If surfaces are going to become a viable tool, they must possess the same operators as other geometry. The transformation for a surface is a minimal requirement. Functions such as trimming a surface or editing a synthetic surface by additions and/or deletions are desirable. This list of functions is almost endless. The point is that today's surfaces have limited usefulness in defining complex parts.

Another problem is trying to visualize 3-D geometry. The capability of view-independent construction is effective only if the designer can determine what he is seeing. The user needs to rotate and zoom in on this component in real time in order to get a feeling for the structure. Depth cueing is needed in order for the designer to visualize the three-dimensionality of the part. For geometric construction this is more desirable than hidden-line removal.

FUTURE NEEDS AND DIRECTION

The ultimate graphics system would be able to define an engine component in such a way that all vital information associated with the component would be easily accessible to all disciplines that must interface with it. In more tangible terms this means that all physical attributes of the part are defined or easily calculated. As a minimum, the part must be fully defined in a 3-D local coordinate system. However, the part also belongs to an assembly and should be referenced by a second 3-D coordinate system that corresponds to the assembly. Since the part may be located in more than one assembly, a bill-of-materials structure is needed to define the multiple occurences. Properties such as material, surface finish, weight, and other mass properties can be determined by interrogating the part.

There exists significant evidence that volumetric modeling will become the next logical progression in computer-aided design. Several commercial CAD systems already offer such capabilities in a limited form. This is consistent with our goal to ultimately eliminate need for 2-D drawings as a document, and to have a single geometric definition for design, analysis, and manufacturing. Whether these goals are to be achieved solely with solids or whether it will take a combination of solids and surfaces is still unknown. Clearly, if solids could address every possible geometric shape in a turbine engine, then volumetric modeling would be an ideal tool.

However, unless a volumetric modeling package is complete in terms of building blocks and operators, the total package is not really useful. Any tool that is made available must be geared towards the production environment. A serious development effort is definitel warranted because of the many benefits that are an outgrowth of a system based on solids. These benefits include fit and interference checking, tolerance accumulation, and volume properties. These are al items critical in the design of turbine engines.

The time frame between conceptual design and first engine testing is rapidly decreasing. To deal with this, group technology coding systems using geometric data bases must be implemented. Interactive computer programs using dynamic display devices will be used to determine the viability of a design in a rapid manner.

By changing critical parameters, the designer will obtain immedia feedback regarding the component's behavior under engine operating conditions. Programs capable of simulating motion under load, dynamic displacement studies and kinematic studies will be used readily to produce highly optimized designs.

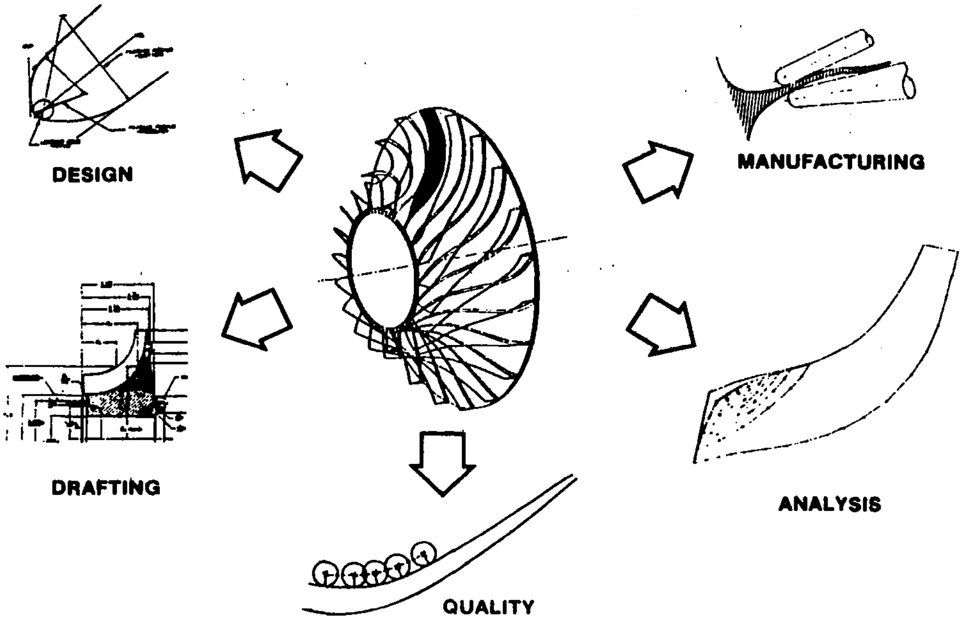


Figure 1.- CAD/CAM system integration.

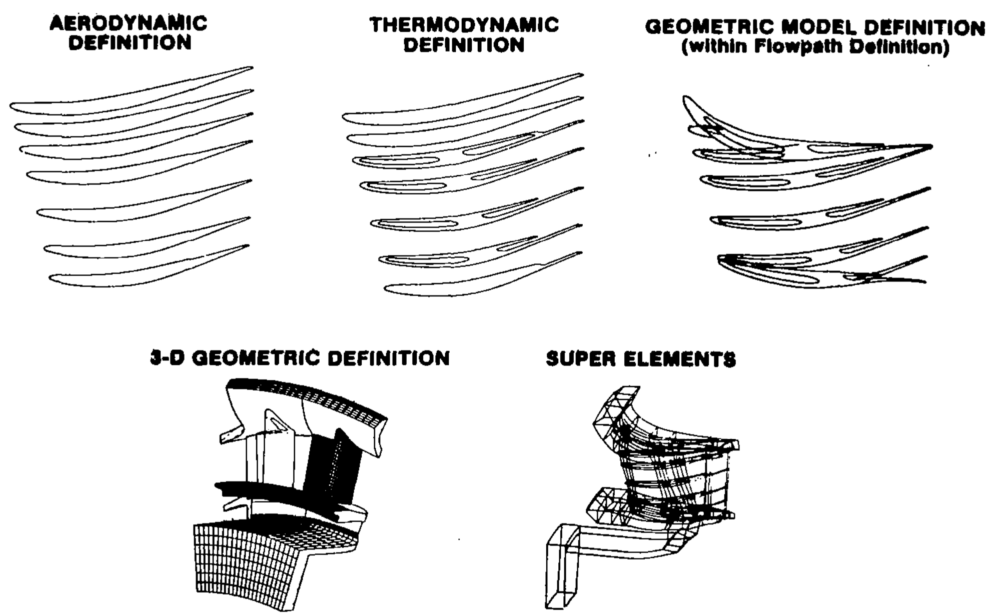


Figure 2.- LT 101 gas producer nozzle.

A Collection of Procedures for Defining Airplane Surfaces for Input to PANAIR

Ralph L. Carmichael
NASA Ames Research Center
Moffett Field, CA

INTRODUCTION

Aerodynamic panel codes, such as PANAIR (Ref. 1), enable researchers and designers to compute the airflow about an airplane configuration. In PANAIR, the geometry of the boundaries is defined by specifying networks of grid points which lie on the surface of the configuration. Each network is a lattice of points, meaning that the points are arranged in rows and columns (Fig. 1). The collection of networks must fit together and cover the entire surface of the aircraft. The PANAIR user is required to generate a large number of (x,y,z) coordinates of these lattices of surface definition points. The generation of these points can be tedious and error-prone and users will probably use some form of automated scheme for generating surface points. One approach for generating these points lies in modifying a computer-aided design (CAD) system to produce networks of surface points in PANAIR format. The purpose of this paper is to describe an alternate approach that enables PANAIR users to define configurations with a collection of procedures that can run on small computer systems. The procedures are based upon the construction of parametric bicubic patches which enable the user to define the external surface of realistic airplane shapes. A complementary program generates a set of points on this continuous surface which is in the format required for input to PANAIR.

DIRECT PANELLING

The first procedure in this collection permits the user to define wings or bodies (or any combination) by specifying various critical coordinates and then to compute the intermediate points by interpolation. For example, to define a trapezoidal wing the critical coordinates are the leading and trailing edges at the root and tip (Fig. 2). The other parameters to be specified are the number of rows and columns, the spacing rule used for interpolation, and the thickness/chord ratio and airfoil section selected from an internal directory of sections. Bodies of revolution are assumed to be made of a nose region, a cylindrical region and an afterbody. The critical coordinates are the length of these regions, radii of the cylindrical region, nose, and base, and the body shape chosen from an internal directory. The number of fuselage stations at which grid points are defined as well as the number of meridian lines must also be specified. The input data shown in figure 3 will produce the simple configuration illustrated.

SURFACE DEFINITION BY PARAMETRIC BICUBIC PATCHES

Direct panelling is very useful for defining simple shapes, but lacks flexibility for producing the complex shapes for which PANAIR was designed. For more general shapes, we define the surface as a collection of parametric bicubic patches (Ref. 2). A parametric bicubic patch is a surface in three dimensional space that is the image of the unit square [0,1]x[0,1] under a bicubic mapping (Fig. 4). After the surface is defined, the PANAIR network points are generated by an interpolation procedure to be described in a later section.

A program has been written which simplifies the process of creating a set of patches. While there are many ways one can specify the data necessary to define the 48 patch coefficients, three have been selected for use. All three are based on the idea of using parametric cubic curves to define the patch (Fig. 5). The first defines a ruled surface made up of straight lines connecting points of equal parametric value on each of two curves. The second connects four curves which make up the four edges of the patch. The third also uses four curves. The first and last curves are opposite edges of the patch, while the other two are intermediate curves lying between them. The patch is required to contain all four curves. Each of these methods suffices to define a patch from the required curves. The curves that make up the patch may be lines, parabolas, circular arcs or cubics (Fig 6). The program requires a command file consisting of keywords followed by the appropriate data. The list of commands and appropriate data is shown in figure 7. An example of the commands to produce a patch representing the quadrant of a cone is shown in figure 8.

SPLINE METHODS

The next procedure to be described enables the user to define a set of patches that fits a shape defined by cross-sections at a number of stations (Fig. 9). Each cross section is defined by a number of points and end conditions that permit a spline to be fitted through these definition points. These are called section splines (Fig. 10). Another set of splines is defined in the opposite direction, the so-called "longitudinal splines" (Fig 11). These are defined through selected points on the section splines. With the lattice of section splines and longitudinal splines as edges, a collection of patches is created which represents a general configuration (Fig 12).

The usual interpolating splines tend to overshoot and are unable to fit sharp corners. The splines used in the program are the nu-splines developed by Neilson (Ref. 3). These splines include a parameter called tension at each point along the spline which controls the local curvature and enables a wider variety of curves to be interpolated (Fig 13). The nu-splines reduce to the usual cubic splines when the tension is set to zero everywhere along the curve.

MODIFICATION OF A SET OF PATCHES

It is frequently necessary to modify a set of patches in order to change an object. Some examples of this are translating or rotating a patch, forcing a patch to connect to another patch (fig. 14), deleting a patch, etc. A utility program has been written to carry out these modifications.

MAKING PANAIR NETWORKS FROM PATCHES

After exercising one or more of the patch generation and modification programs, a user eventually creates a collection of patches which represents the desired configuration. A program has been written which creates PANAIR network files from a file of patch coefficients and a command file called a network definition file (Fig. 15). The network definition file contains the numbers of the patches used to make a network, the number of grid points to be interpolated and the spacing rule to be used.

MODIFICATION OF A SET OF NETWORKS

It is frequently necessary to modify a PANAIR network file. For instance, a network needs to be translated, deleted, or connected to another network. Another utility program has been developed which allows these functions to be carried out (Fig 16). Commands are available that allow two networks to be merged into one or allow one network to be split into two. There are commands for attaching base networks or wake networks to existing networks.

AN EXAMPLE OF A COMPLETE MODEL DEFINITION

To clarify some of the concepts shown here, an example will be worked to show exactly what is required to create a file that may be used as the network definition for a PANAIR run. The wing-body configuration is illustrated in figure 17, taken from reference 4. The body is made up of a nose followed by a cylindrical (but not circular) section. The thin wing has a parabolic section. The definition of the wing is illustrated in figure 18, the cylindrical section of the body in figure 19 and the nose in figure 20. The resulting 9 patches define the upper right quadrant of the model (Fig 21). Figure 22 illustrates the use of the patch modification program to force a connection between the wing and body patches, to make images of the patches across the z=0 plane, and to create a new patch that closes the wing tip. Networks are created from the revised patches (Fig 23) and the network definition file, as shown in figure 24, and the resulting networks are illustrated in figure 25. The remaining networks are created from the network modification routine as illustrated in figure 26. The revised networks are shown in figures 27 and 28 for the right half of the vehicle, and both sides are shown in figure 29.

Figure 30 illustrates a complex configuration that has been developed with the techniques described here. The fuselage was defined by splines (SURFCEM), while the nacelles, wings, and tails were made with MAKEPATCH. The patch and network modification programs were also used to develop this configuration.

CONCLUSIONS

In order to produce networks for PANAIR, the user is greatly aided by a geometry package that will produce networks in the correct format. Although the best solution to this problem is the use of a suitably modified computer-aided design system, a collection of programs has been written to allow the user with a small-scale computer to generate networks that represent configurations of considerable generality (Fig. 31). This repertoire of programs enables users of PANAIR to create networks with a minimum of effort and should increase the overall usefulness of the PANAIR program.

REFERENCES

1. Carmichael, Ralph L., and Erickson, Larry L.: "PANAIR - A Higher Order Panel Method for Predicting Subsonic and Supersonic Linear Potential Flows About Arbitrary Configurations," AIAA Paper 81-1255, June 1981.

2. Peters, George J.: "Interactive Computer Graphics Application of the Parametric Bi-cubic Surface to Engineering Design Problems". In "Computer Aided Geometric Design", Barnhill and Riesenfeld, ed., Academic Press, 1974.

3. Neilson, Gregory M.: "Some Piecewise Polynomial Alternatives to Splines Under Tension". In "Computer Aided Geometric Design", Barnhill and Riesenfeld, ed., Academic Press, 1974.

4. Gloss, Blair B.: "Effect of Canard Location and Size on Canard-Wing Interference and Aerodynamic Center Shift Related to Maneuvering Aircraft at Transonic Speeds." NASA TN D-7505, June 1974.

```
NETWORK= FUSELAGE                    5        6
 0.0  0.0  0.0      0.0  0.0    0.0     0.0  0.0  0.0
 0.0  0.0  0.0      0.0  0.0    0.0
 4.0  0.0  .75      4.0   .53    .53    4.0  .75  0.0
 4.0  .53 -.53      4.0  C.0    -.75
 8.0  0.0  1.0      8.0   .707   .707   8.0  1.0  0.0
 8.0  .707 -.707    8 0  0.0   -1.0
 12.  0.0  1.0     12.0   .707   .707  12.0  1.0  0.0
 12.  .707 -.707   12.0  0.0   -1.0
 16.  0.0  1.0     16.0   .707   .707  16.0  1.0  0.0
 16.  .707 -.707   16.0  0.0   -1.0
 20.  0.0  1.0     20.0   .707   .707  20.0  1.0  0.0
 20.  .707 -.707   20.0  0.0   -1.0
BOUNDARY CONDITION=1 UPPER
NETWORK= WING           4       5
  8.0  1.0   0.0  12.   1.0   0.0
  16.  1.0   0.0  20.   1.0   0.0
 11.0  3.25  0.0  14.0  3.25  0.0
 17.0  3.25  0.0  20.0  3.25  0.0
 14.0  5.5   0.0  16.0  5.5   0.0
 18.0  5.5   0.0  20.0  5.5   0.0
 17.0  7.75  0.0  18.0  7.75  0.0
 19.0  7.75  0.0  20.0  7.75  0.0
 20.0  10.0  0.0  20.0  10.0  0.0
 20.0  10.0  0.0  20.0  10.0  0.0
BOUNDARY CONDITION=1 AVERAGE
```

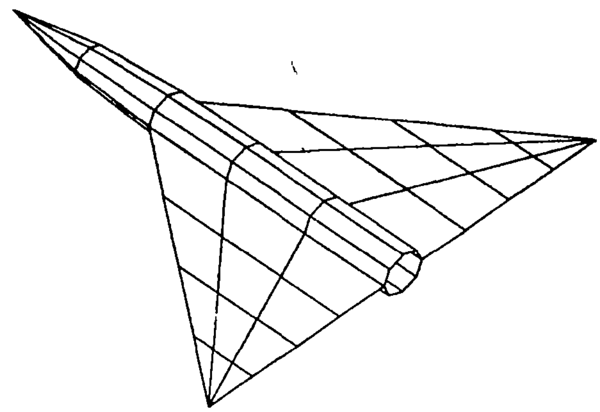


Figure 1.- PANAIR input.

WING

ROOT	TIP	
XLE	XLE	
XTE	XTE	
Y	Y	
Z	Z	
SECTION	SECTION	000x, 6400X, 6500X, parabolic, double-wedge
T/C	T/C	

BODY

RADIUS

LNOSE

LAFTERBODY

LBODY

SHAPE: cone, parabolic, Sears-Haack, von Karman

(GEOMWBODY)

Figure 2.- Direct panelling of simple shapes.

```
$BODY   LNOSE=8, LBODY=20,
  RADIUS=1, SHAPE=1,
  NFS=6, NTHETA=5,
  NETNAME='FUSELAGE' $
$WING   ROOT=8,20,1,0,
  TIP=20,20,10,0,
  ROWS=3, COLS=4, THICK=0,0,
  NETNAME='WING' $
```

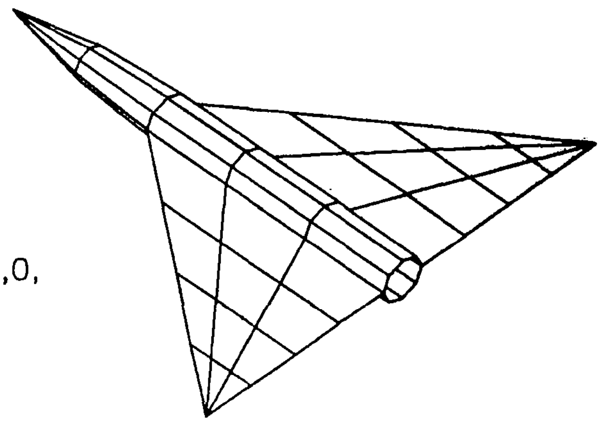
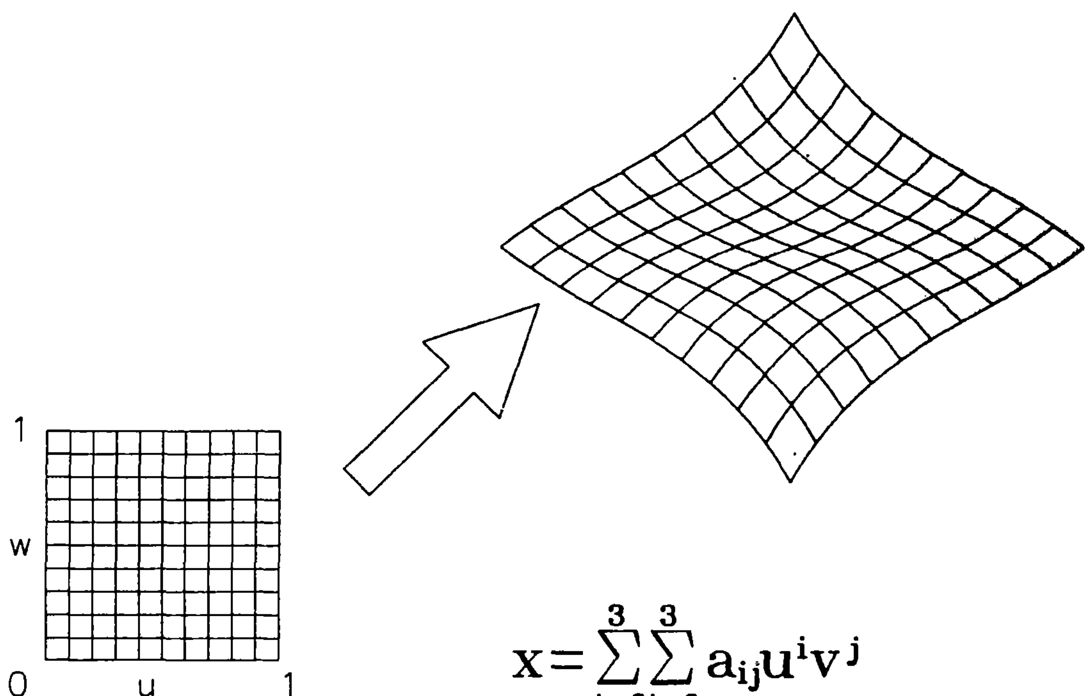


Figure 3.- Sample input.



$$x = \sum_{i=0}^{3} \sum_{j=0}^{3} a_{ij} u^i v^j$$

Figure 4.- Parametric bicubic patches.

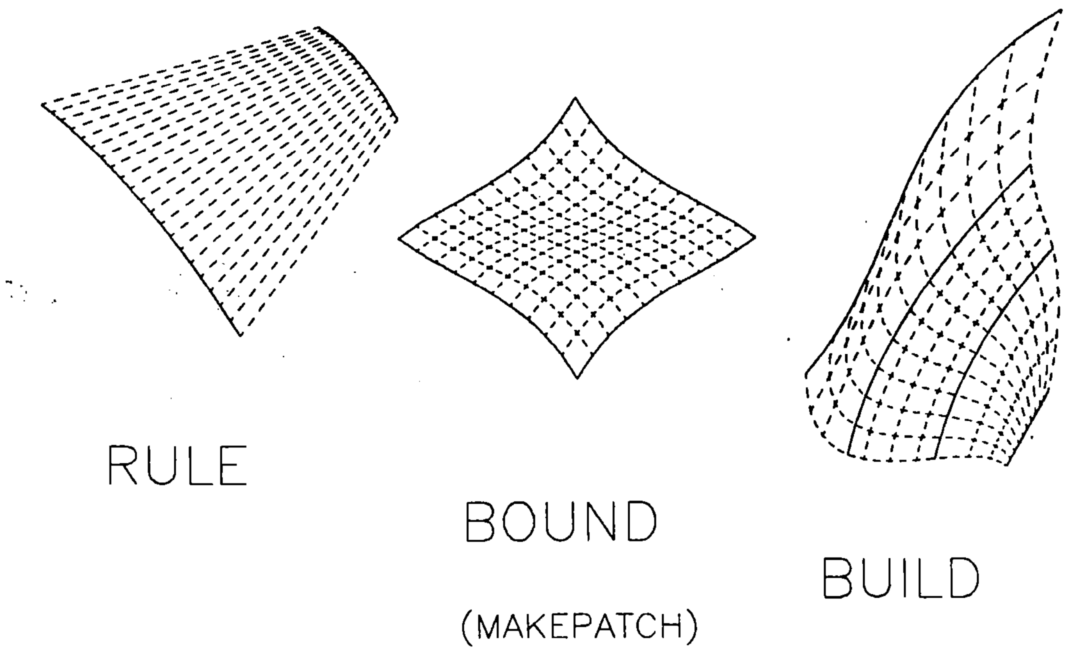


Figure 5.- Making patches from curves.

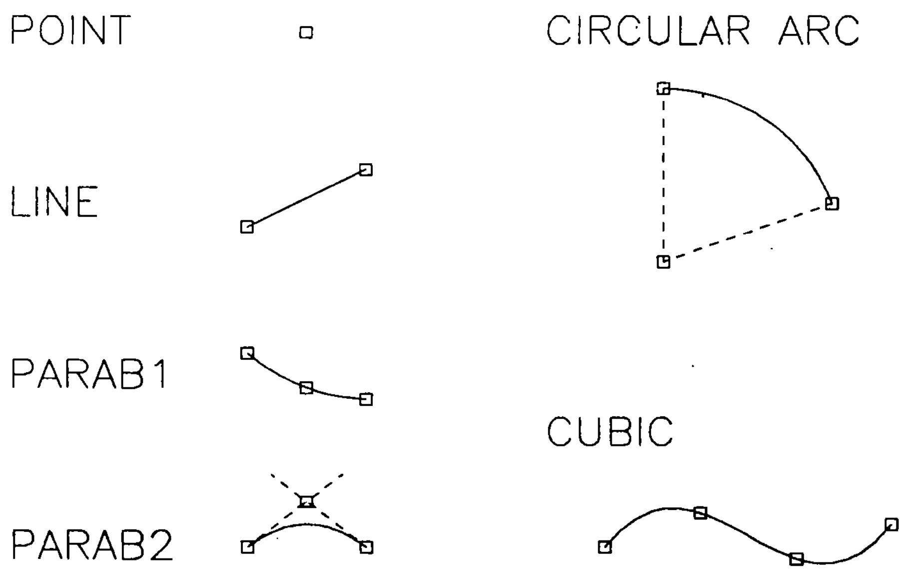


Figure 6.- Curve definition.

PATCHES

RULE
(two curves)

BOUND
(four curves)

BUILD
(four curves)

CURVES

POINT
x y z

LINE
x y z (1st endpoint)
x y z (2nd endpoint)

CIRCLE (arc)
x y z (1st endpoint)
x y z (2nd endpoint)
x y z (center)

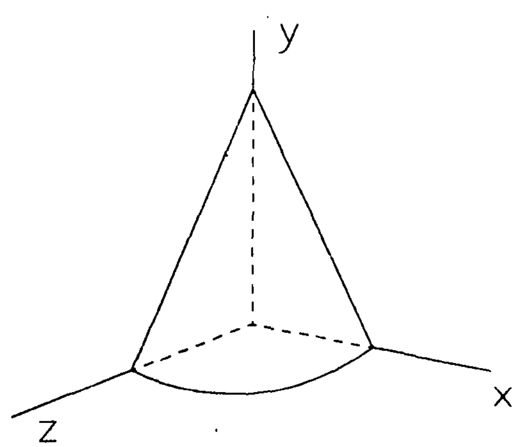
PARAB1
x y z (1st endpoint)
x y z (2nd endpoint)
x y z (midpoint)

PARAB2
x y z (1st endpoint)
x y z (2nd endpoint)
x y z (intersection)

CUBIC
x y z (1st point)
x y z (2nd point)
x y z (3rd point)
x y z (4th point)

Figure 7.- Commands for MAKEPATCH.

RULE
POINT
0 1 0
CIRCLE
0 0 1
1 0 0
0 0 0



defines a quadrant of a cone

Figure 8.- Making a patch.

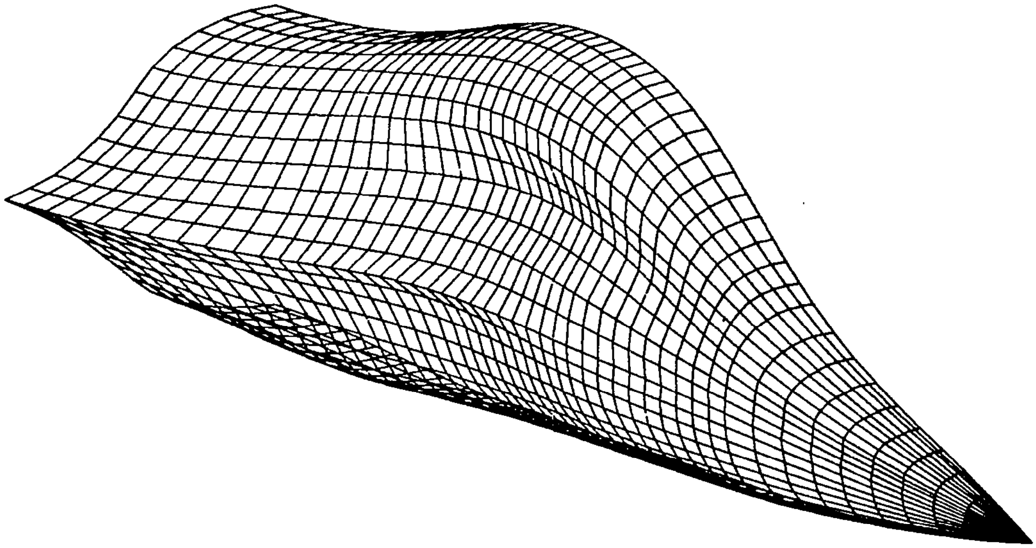


Figure 9.- Spline methods.

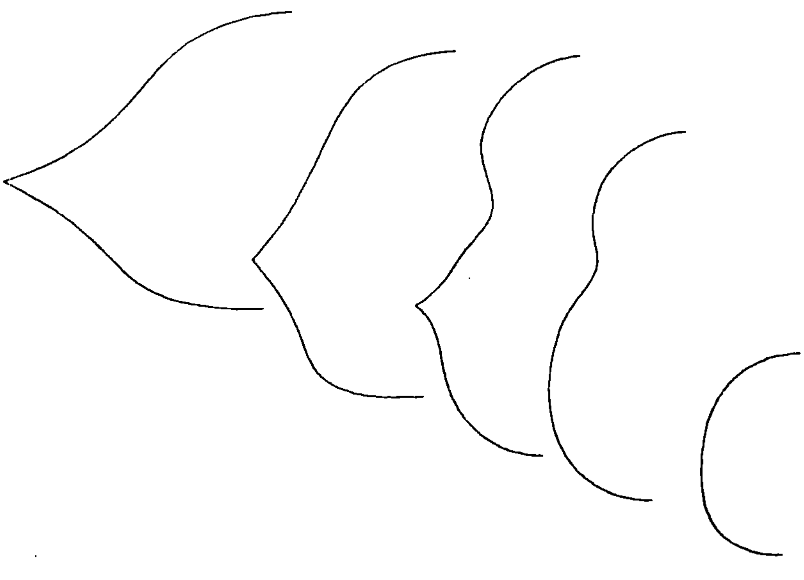


Figure 10.- Cross sections.

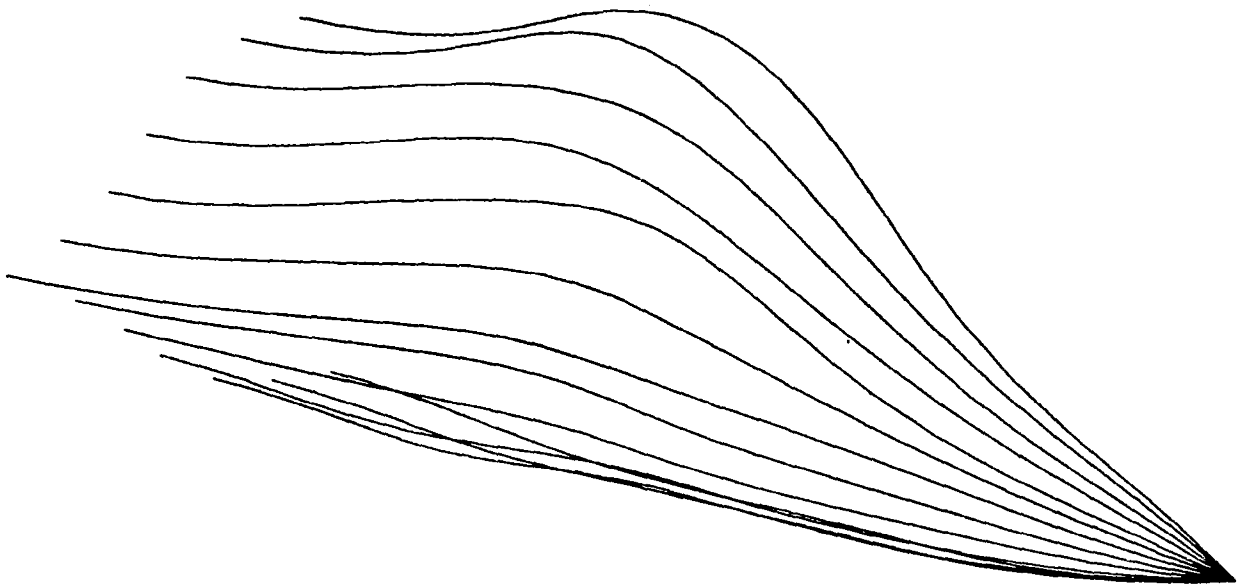


Figure 11.- Longitudinal members.

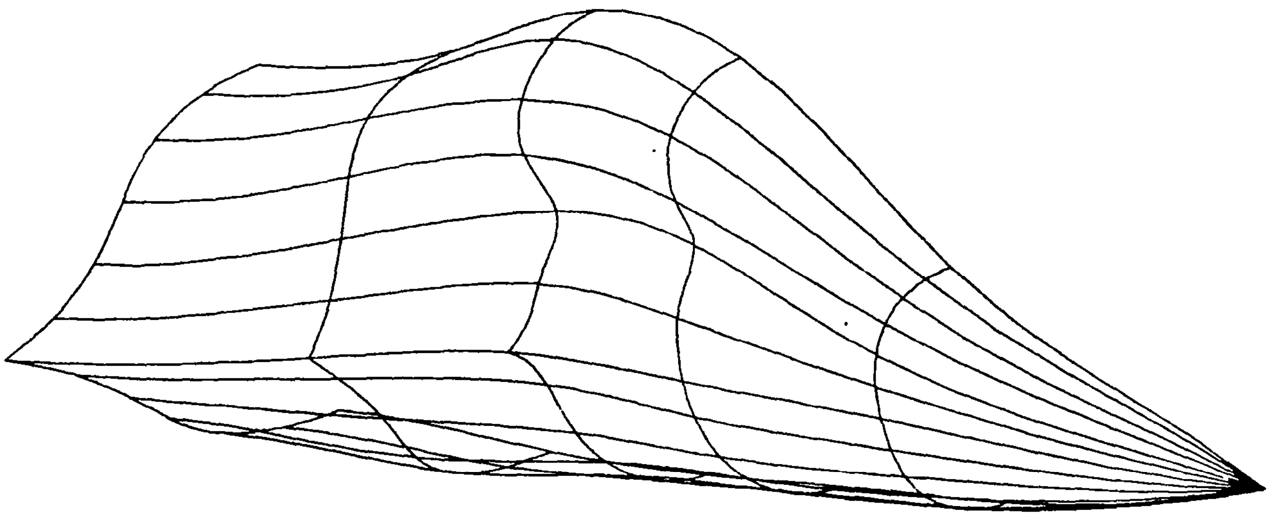


Figure 12.- Patch edges.

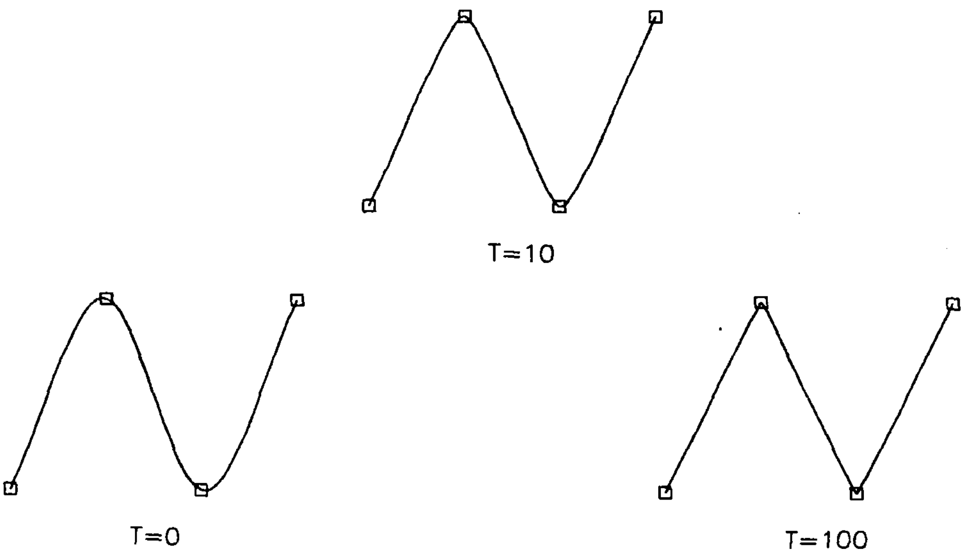


Figure 13.- The effect of tension.

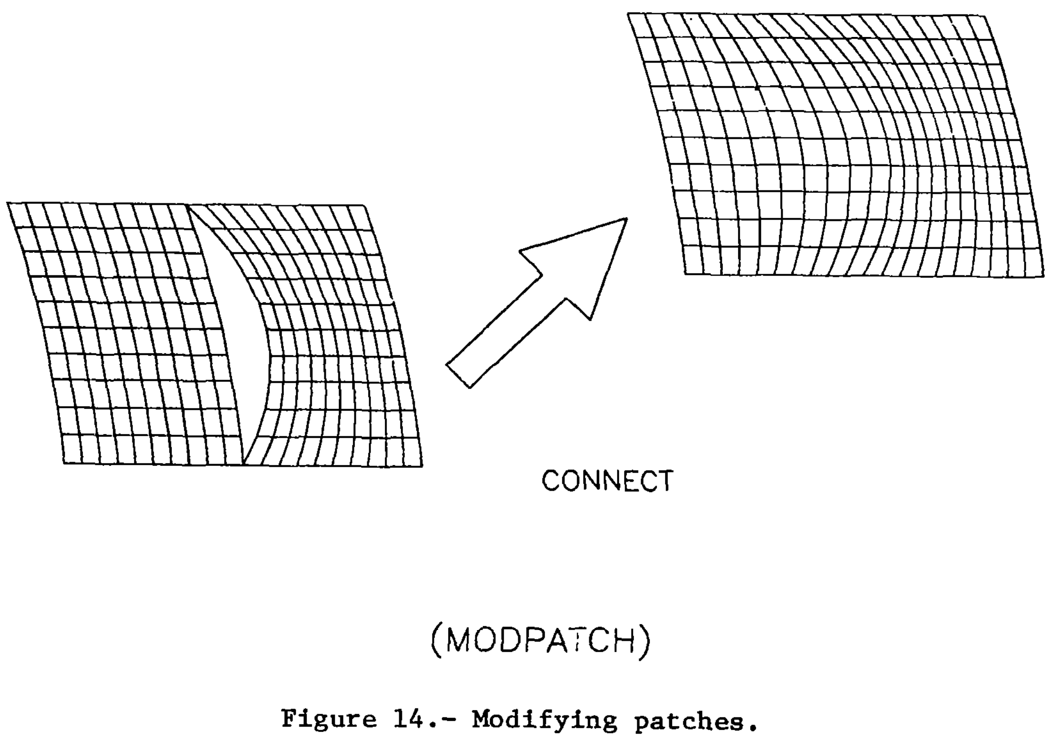


Figure 14.- Modifying patches.

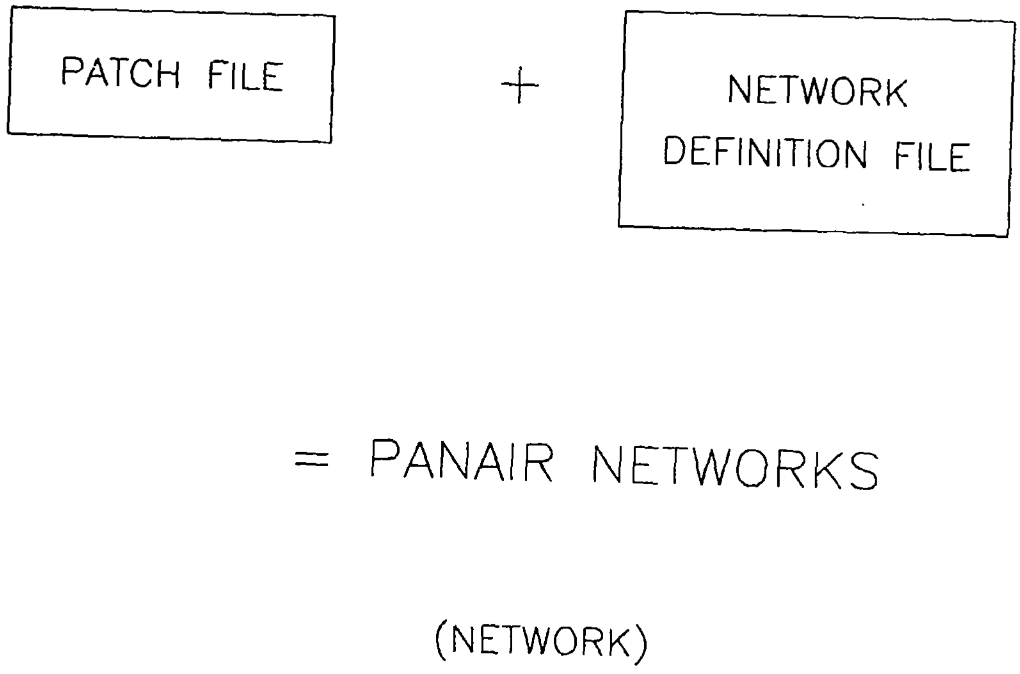


Figure 15.- Making networks.

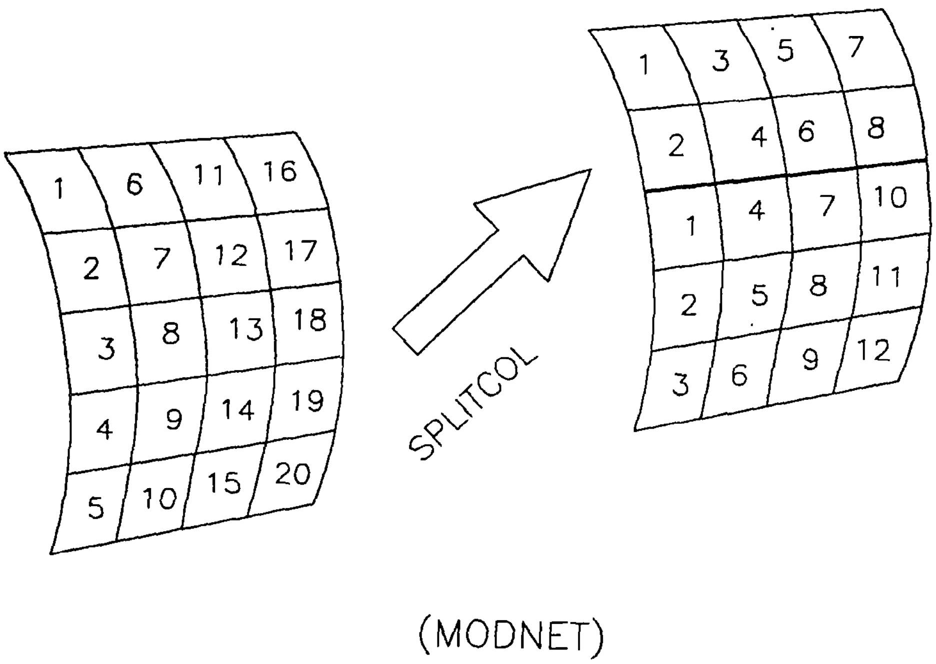


Figure 16.- Modifying networks.

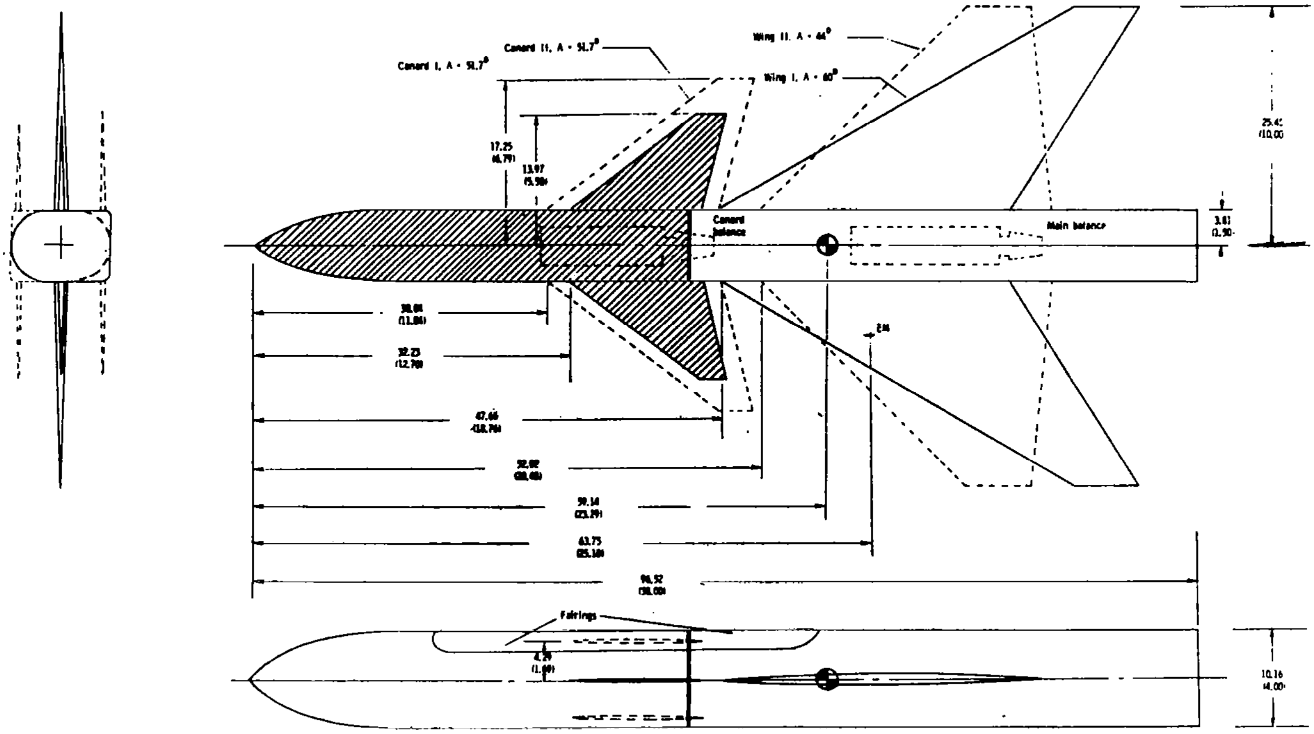


Figure 17.- An example. (All dimensions in centimeters (inches).) (From ref. 4.)

RULE

PARAB1

47.65 3.81 0

77.45 3.81 0

62.55 3.81 .894

PARAB1

85.04 25.4 0

91.81 25.4 0

88.425 25.4 .1354

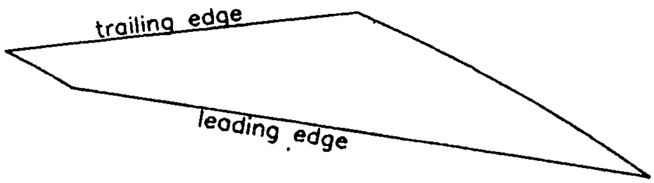


Figure 18.- MAKEPATCH input for wing.

```
RULE
   CIRCLE
      14      0   5.08
      14  3.81   1.27
      14      0   1.27
   CIRCLE
      47.65      0   5.08
      47.65  3.81    1.27
      47.65      0   1.27
RULE
   LINE
      14  3.81  1.27
      14  3.81      0
   LINE
      47.65  3.81  1.27
      47.65  3.81  0
```

(repeat for 47.65 to 77.45 and 77.45 to 96.82)

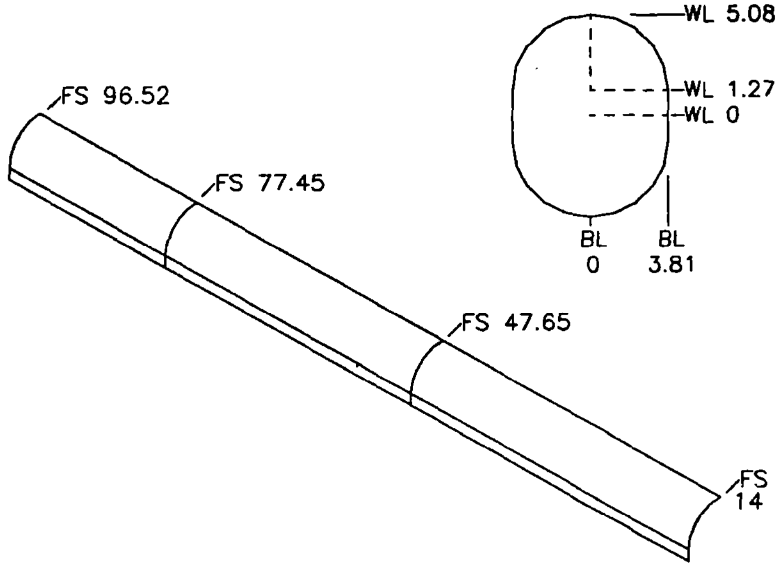


Figure 19.- MAKEPATCH input for cylindrical part of body.

```
BOUND
   POINT
      0 0 0
   PARAB2
      0     0     0
      14  3.81  1.27
      7  3.81  1.27
   CIRCLE
      14     0  5.08
      14  3.81  1.27
      14     0  1.27
   PARAB2
      0  0     0
      14  0  5.08
      7  0  5.08
```

```
BOUND
   POINT
      0 0 0
   PARAB2
      0    0   0
      14  3.81  0
      7  3.81  0
   LINE
      14  3.81  1.27
      14  3.81  0
   PARAB2
      0     0     0
      14  3.81  1.27
      7  3.81  1.27
```

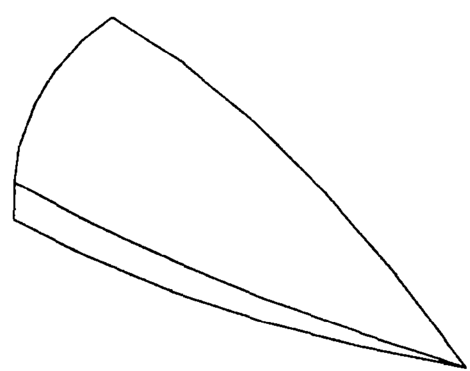


Figure 20.- MAKEPATCH input for nose.

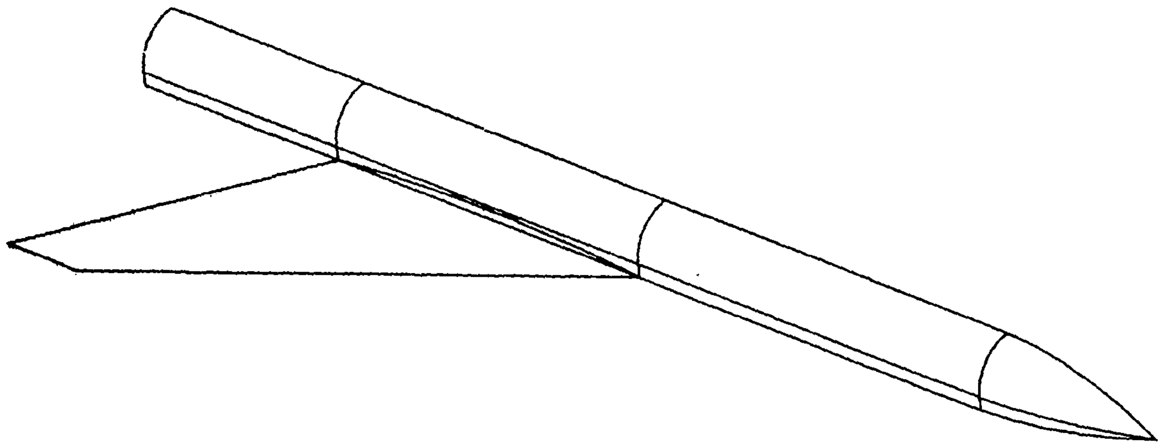


Figure 21.- Patches generated.

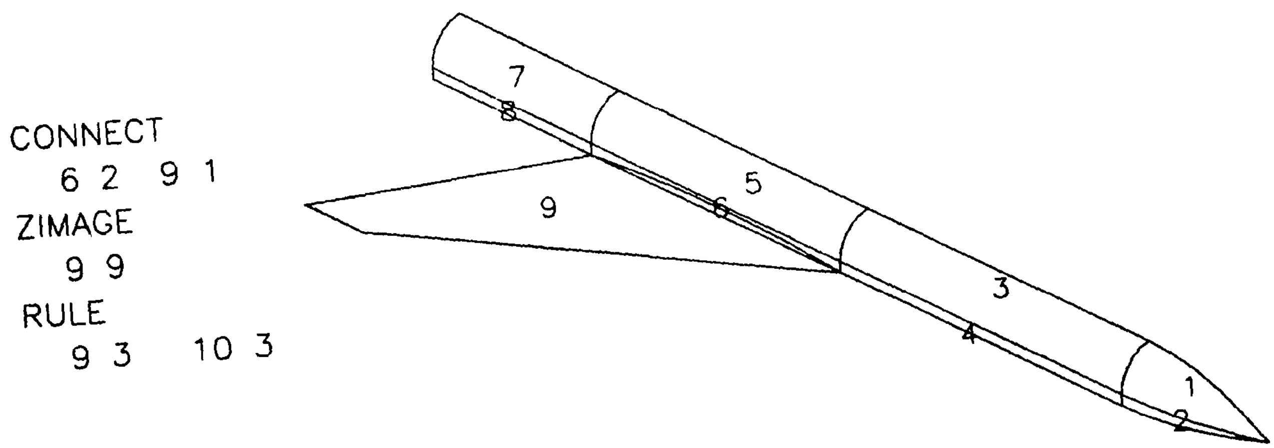


Figure 22.- MODPATCH input.

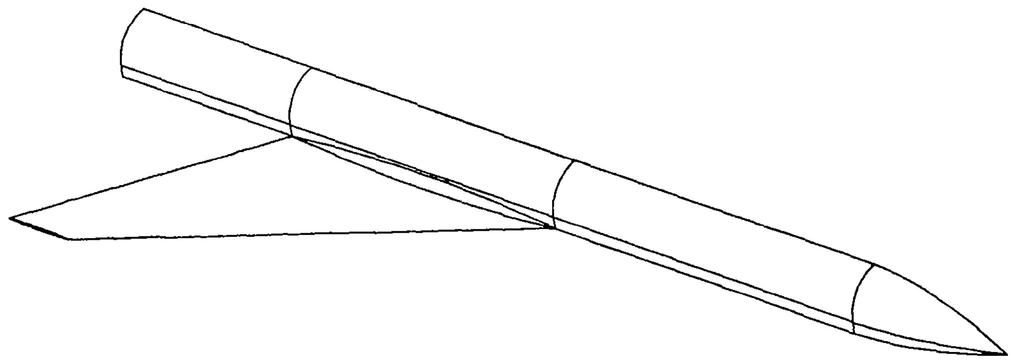


Figure 23.- Revised patches.

```
$NETWORK  NETNAME='TOP-NOSE(1)',
     NLE=2, NSE=2, IPATCH=1,3,2,4,
     ICLASS=1, ISUBCL=1,
     ILESP=1, ISESP=1, ILEPT=5, ISEPT=5$
$NETWORK  NETNAME='TOP-MID-FUS(2)',
     NLE=2, NSE=1, IPATCH=5,6, LEPAN1=1, ISEPT=4$
$NETWORK  NETNAME='TOP-AFT-FUS(3)', IPATCH=7,8, ISEPT=3$
$NETWORK  NETNAME='TOP-WING(4)', NLE=1, NSE=1,
     LEPAN1=4, SEPAN1=1, IPATCH=9, ILEPT=9, ISEPT=4$
$NETWORK  NETNAME='TIP-WING(5)', NLE=1, NSE=1,
     LEPAN1=4, SEPAN1=1, IPATCH=11, ILEPT=3, ISEPT=4$
```

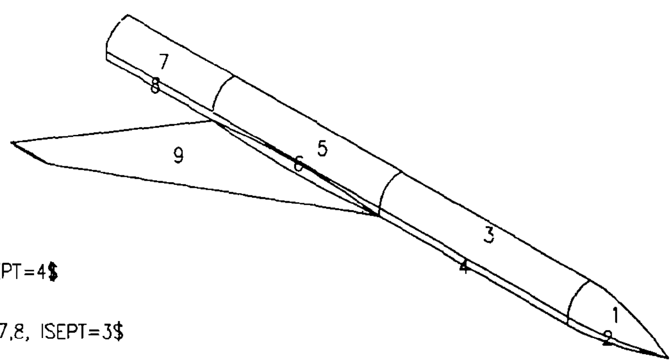


Figure 24.- Network definition file.

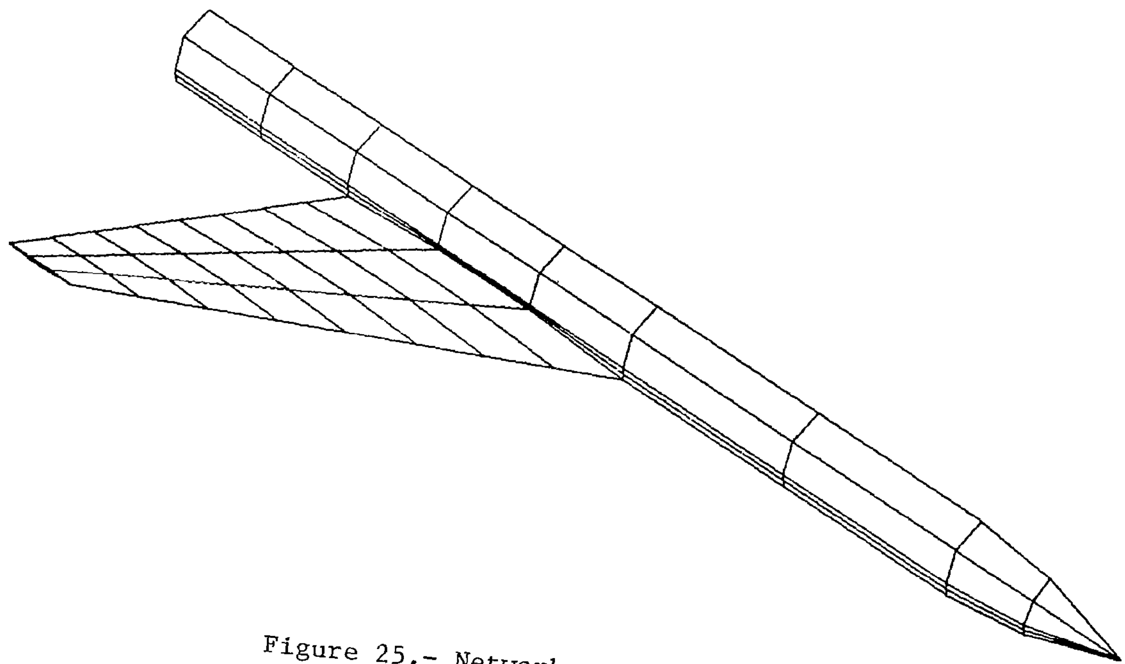


Figure 25.- Networks generated.

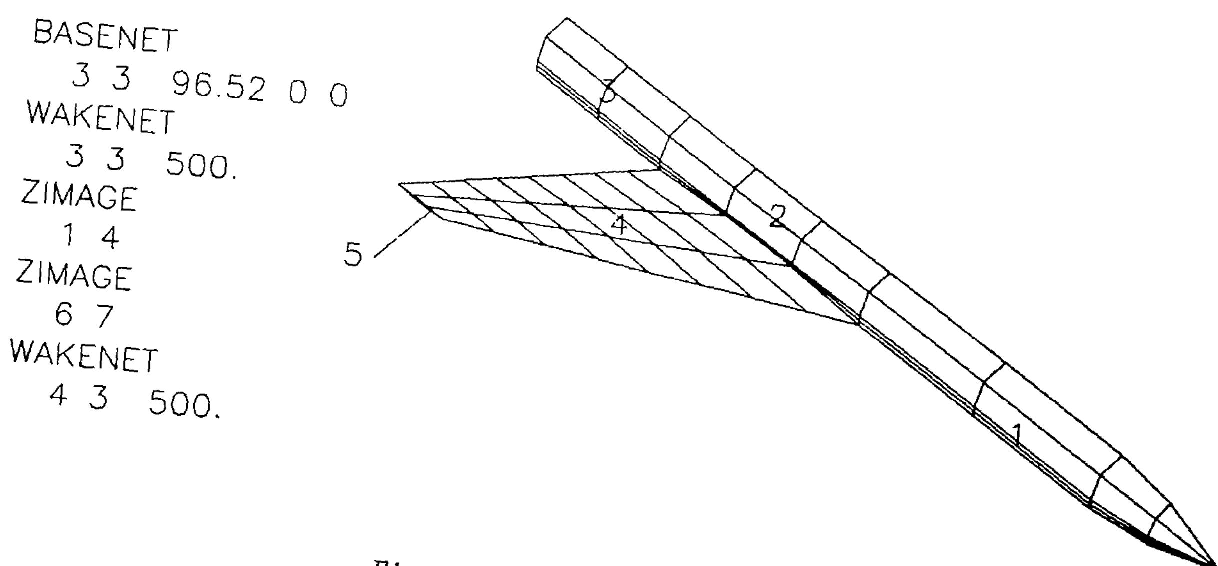


Figure 26.- MODNET input.

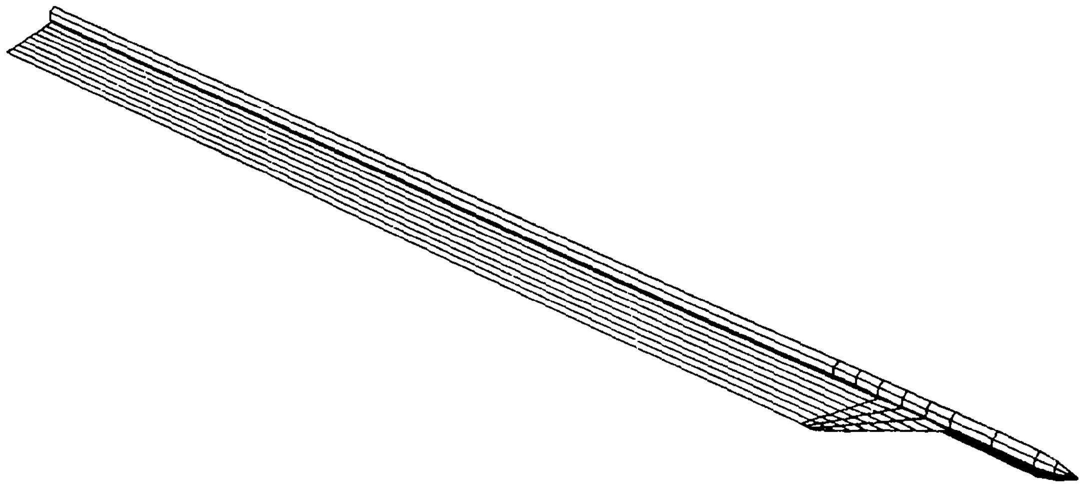


Figure 27.- Revised networks.

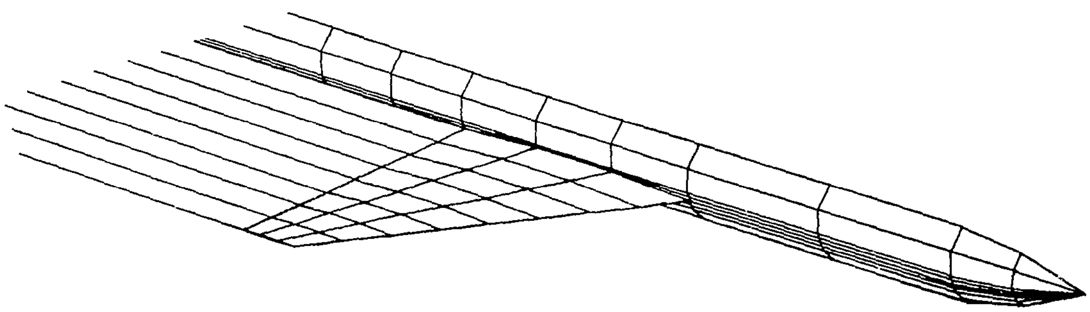


Figure 28.- Revised networks.

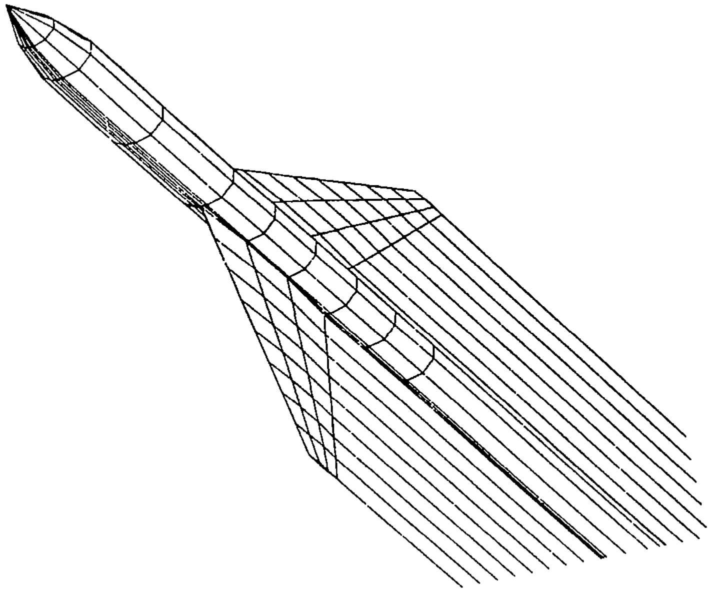


Figure 29.- Complete configuration.

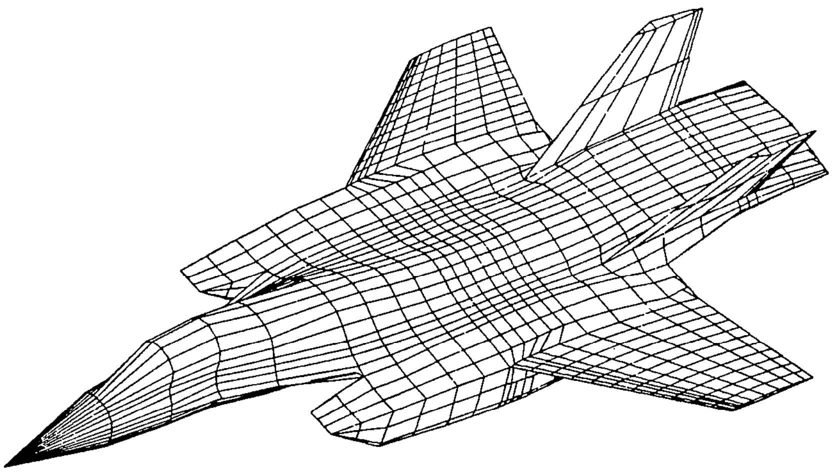


Figure 30.- Complex configuration.

GEOMWBODY

MAKEPATCH

MODPATCH

NETWORK

MODNET

PANAIR

Figure 31.- Summary of programs.

THE COMPUTATION OF MESH CONFIGURATIONS FOR THREE-DIMENSIONAL FLOW ANALYSIS

S. G. Gibson
Boeing Commercial Airline Company
Seattle, Washington

A geometry modeling method (Reference 1) has been developed to define three-dimensional mesh configurations, which are expressed by the set of surface intersections with a coordinate mesh. These configurations define bodies for transonic potential-flow computation (Reference 2) in simple physical coordinates.

A computational mesh is defined by a set of values for each coordinate. Mesh lines are placed where two coordinates are fixed at mesh values and the third coordinate varies. The boundary condition at solid surfaces is determined by the position and the surface normal direction at each intersection of a mesh line with the surface. A typical configuration uses mesh sets of up to 80 values for each coordinate, giving on the order of 5000 mesh-surface intersections.

Such configurations require assistance for effective preparation. A geometry modeling method with the flexibility that the computational method gives must be provided. The large number of intersections requires reliable preparation to give a consistent configuration. Also, the modeling method must represent the geometry accurately. The geometry modeling method presented in this paper has been developed to prepare the required mesh configurations.

The surface is modeled independently of the mesh, as a set of parametric bicubic patches. The independence of surface and mesh representation minimizes the effort needed to vary the mesh used with a configuration.

Surface regions are described by networks of section curves and crosswise members. A knot-selector program accepts a family of input curves, interpolates new points on them, and completes the surface description. The new points are selected with spacings that concentrate them according to the curvature distribution on the surface.

Parameters are assigned to the section and member curves proportionately to the accumulated chord lengths along them. Each patch is defined by surface information at its four corners: position, sectionwise tangent, memberwise tangent, and twist vectors. Derivatives are defined by fitting parametric cubic splines to the curves. Twist values are defined by fitting the sectionwise tangent values along each member with a spline. The parametric region for each patch is transformed to a unit square by scaling the derivatives.

Surface models must be verified to show that they model the desired configurations and that their regions join smoothly. This is done with the aid of the graphics displays Shaded graphics show the physical geometry while hiding the parametric representation used. Wire frame models are also available to visualize parametric details.

Mesh-surface intersection normals are computed in a two-step process: plane-surface intersections are interpolated by parametric cubic curves and then these curves are cut by other planes.

Plane-patch intersection is performed by a code that analyzes the bicubic patch for degeneracies and boundary intersections. These divide the patch into strips across which the number of intersections is constant. An input tolerance controls the density of points along the intersections, to bound the interpolation error.

Due to the large number of normals in a configuration, reliability is a key requirement. The reliability of the mesh intersection computation is limited by the approximation of the intersection curve by parametric cubic segments between discrete points. Grazing intersections are an ill-conditioned computation: they can move a normal across a mesh value from the interval where consistency with other normals requires it to be found. Such inconsistencies are detected by a preliminary check in the potential-flow code. A graphic display of the mesh lines and intersections aids users in locating the source of the problem.

A simple program computes a replacement for a normal by interpolation between a surrounding pair.

The flow code lists a table of areas for the duct-like parts of a configuration. A comparison of this table with the configuration definition is a powerful test for correctness of the configuration.

The basic approach is extended by the availability of cylindrical coordinates, surface description transformations between coordinates, and an interface to output a dense set of curves covering surface models. Cylindrical coordinates are used to exploit the character of nearly axisymmetric configurations. Coordinate transformations on surface-description files are implemented. This allows surface regions to be described in local coordinates which exploit their natural symmetry. An interface has been developed to evaluate a dense set of constant-parameter curves on a rectangular patch network. This interface can send surface data to other systems, such as numerical-control shops which machine-test models, and to integrated CAD/CAM networks.

Often configurations are defined with component surfaces extending until another component is intersected. The model of such a surface must be trimmed to exclude mesh intersections from a surface location that lies within the other component.

Various configurations have been prepared for 3-D potential flow analysis, including many bare inlet models in cylindrical coordinates. Trimmed surface models have been used to prepare nacelle installation configurations. A variety of ducts and mixers has been represented. A study has compared analytically defined mesh/surface intersections with similar data obtained from the corresponding surface model. Another study reproduced parameters for an analytic surface definition from mesh-surface intersections obtained from a model of the surface. In both studies the modeling accuracy was shown to be compatible with the flow analysis requirements.

This technique fulfills the requirements of the associated three-dimensional potential-flow analysis. It preserves the flow analysis method's superior flexibility for adapting to unusual geometries. The geometry modeling technique is supported by a sufficient set of tools for effective use by general users, and sufficient accuracy has been demonstrated.

REFERENCES

1. Reyhner, T. A., "Computation of Transonic Potential Flow About Three-Dimensional Inlets, Ducts, and Bodies," NASA CR-3514, March 1982.

2. Gibson, S. G., "User's Manual for MASTER: Modeling of Aerodynamic Surfaces by Three-Dimensional Explicit Representation," NASA CR-166056, January 1983.

3-D MESH CONFIGURATIONS SPECIFY SURFACES FOR TRANSONIC FLOW ANALYSIS

THEIR COMPUTATION IS COMPLICATED AND SENSITIVE

- EACH COORDINATE HAS A DISCRETE SET OF MESH VALUES (TYPICALLY 80)
- HOLDING TWO COORDINATES AT MESH VALUES DEFINES A LINE
- MESH LINES ARE BOUNDED BY THEIR INTERSECTIONS WITH SURFACES
- INTERSECTION LOCATIONS (WITH SURFACE NORMALS) DEFINE A MESH CONFIGURATION
- TYPICAL MESH CONFIGURATIONS CONTAIN 5000 INTERSECTION NORMALS
- THE COMPUTATION OF GRAZING INTERSECTIONS IS ILL-CONDITIONED

SUPPORT OF AERODYNAMIC ENGINEERING ADDS PRACTICAL REQUIREMENTS

MASTER WAS DEVELOPED TO PREPARE PROPULSION CONFIGURATIONS FOR FLOW ANALYSIS

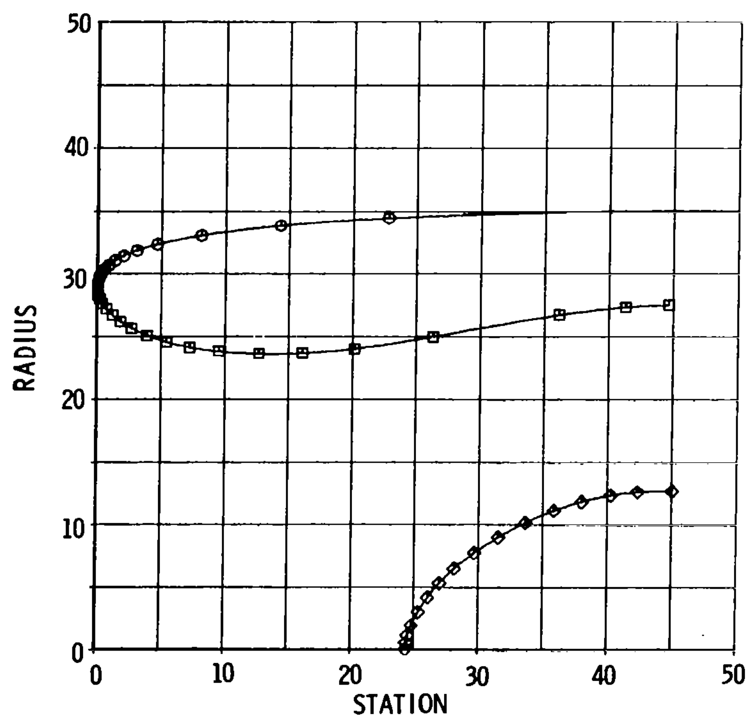
- NON-ANALYTIC GEOMETRIES MUST BE ALLOWED
- MODELING MUST NOT INTRODUCE CURVATURE DISCONTINUITIES
- CYLINDRICAL-COORDINATE MESHES MUST BE ALLOWED
- NORMALS FROM INTERSECTING OBJECTS WITHIN EACH OTHER MUST BE REMOVED
- SYSTEM OPERATION BY NON-EXPERTS MUST BE POSSIBLE
- MESH LOCATIONS SHOULD BE ADJUSTABLE WITHOUT STARTING OVER

A KNOT SELECTOR PREPROCESSES SURFACE DESCRIPTIONS

USERS CAN IGNORE MODELING DETAILS

- RAW SECTION CURVES ARE INPUT
- NEW SECTION POINTS ARE INTERPOLATED: AN EQUAL NUMBER, SIMILARLY SPACED
- POINT DENSITY IS PROPORTIONAL TO CURVATURE
- REGULAR CROSS-MEMBER AND PATCH CORNER DESCRIPTIONS ARE CREATED
- SURFACE DESCRIPTIONS CAN BE EDITED TO CHANGE MODELING DETAILS

KNOT SELECTOR OUTPUT: CURVATURE CONTROLLED SPACING



SURFACES ARE MODELED BY SETS OF PARAMETRIC BICUBIC PATCHES

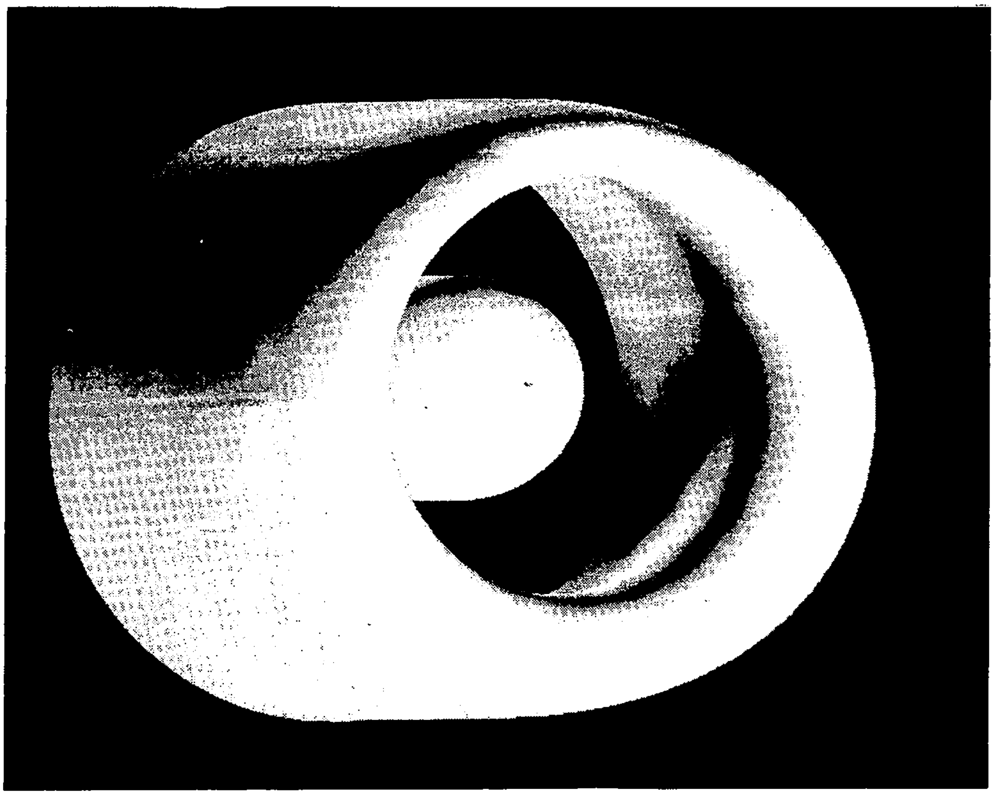
A TRUE THREE-DIMENSIONAL SURFACE REPRESENTATION

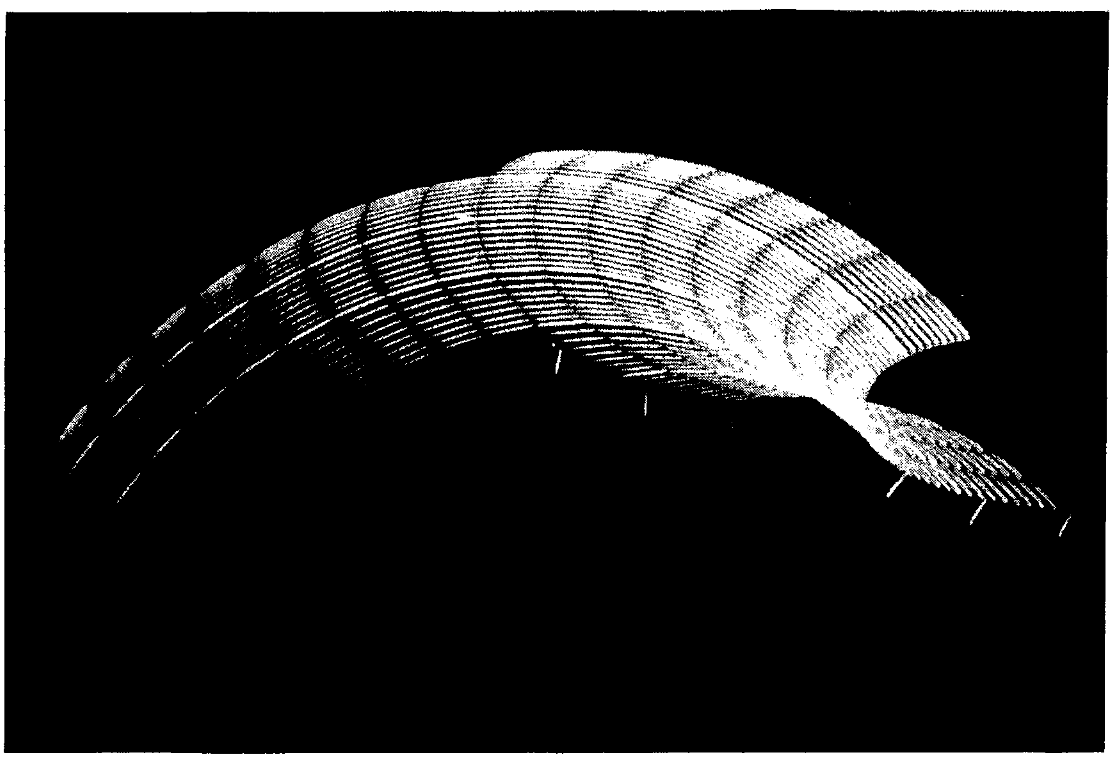
- GEOMETRIC PATCH REPRESENTATION: CONTAINS CORNER POSITIONS AND DERIVATIVES
- CHORD LENGTH PARAMETERIZATION ALONG SECTIONS AND MEMBERS
- SPLINE FITS OF POSITIONS COMPUTE SECTION AND MEMBER TANGENTS
- SPLINE FITS OF SECTION TANGENTS ALONG MEMBERS COMPUTE TWISTS

GRAPHIC DISPLAYS ARE USED TO CHECK SURFACE MODELS

SHADED AND WIRE-FRAME DISPLAYS HAVE COMPLIMENTARY APPLICATIONS

- SHADED DISPLAYS DIRECTLY SHOW SURFACE NORMALS
- SHADED DISPLAYS HIDE PATCH REPRESENTATION DETAILS
- SHADED DISPLAYS REMOVE HIDDEN-SURFACE CLUTTER
- WIRE FRAME DISPLAYS SHOW PATCH REPRESENTATION DETAILS
- WIRE FRAME DISPLAYS CAN COMPARE PATCHES AND INTERSECTION NORMALS





MESH EXTRACTION IS THE MOST COMPLICATED CALCULATION

EXTRACTIONS ARE CALCULATED IN TWO STEPS

STEP 1 (PLANE-PATCH INTERSECTION): COMPUTES CURVES WITH SURFACE NORMAL DATA

STEP 2 (PLANE-CURVE CUTTING): COMPUTES NORMALS ON INTERSECTION CURVES

THE FLOW ANALYSIS CODE CHECKS MESH CONFIGURATIONS FOR CONSISTENCY

DUCT AREA DISTRIBUTIONS ARE USED TO CHECK CONFIGURATIONS

CYLINDRICAL-COORDINATE MESH CONFIGURATIONS CAN BE EXTRACTED

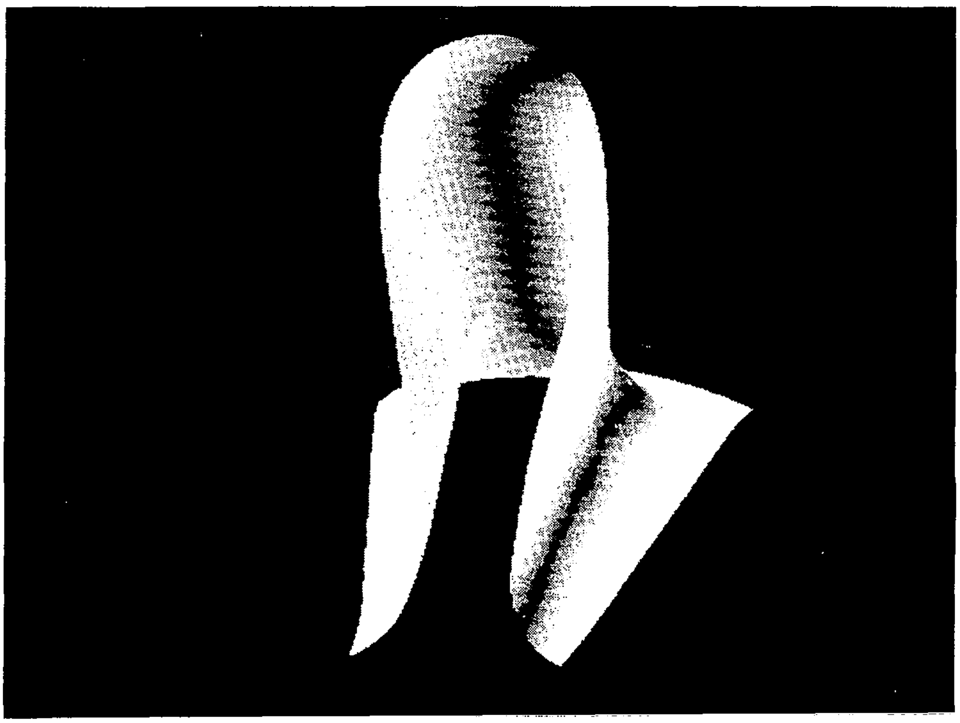
NATURAL SYMMETRY CAN BE USED

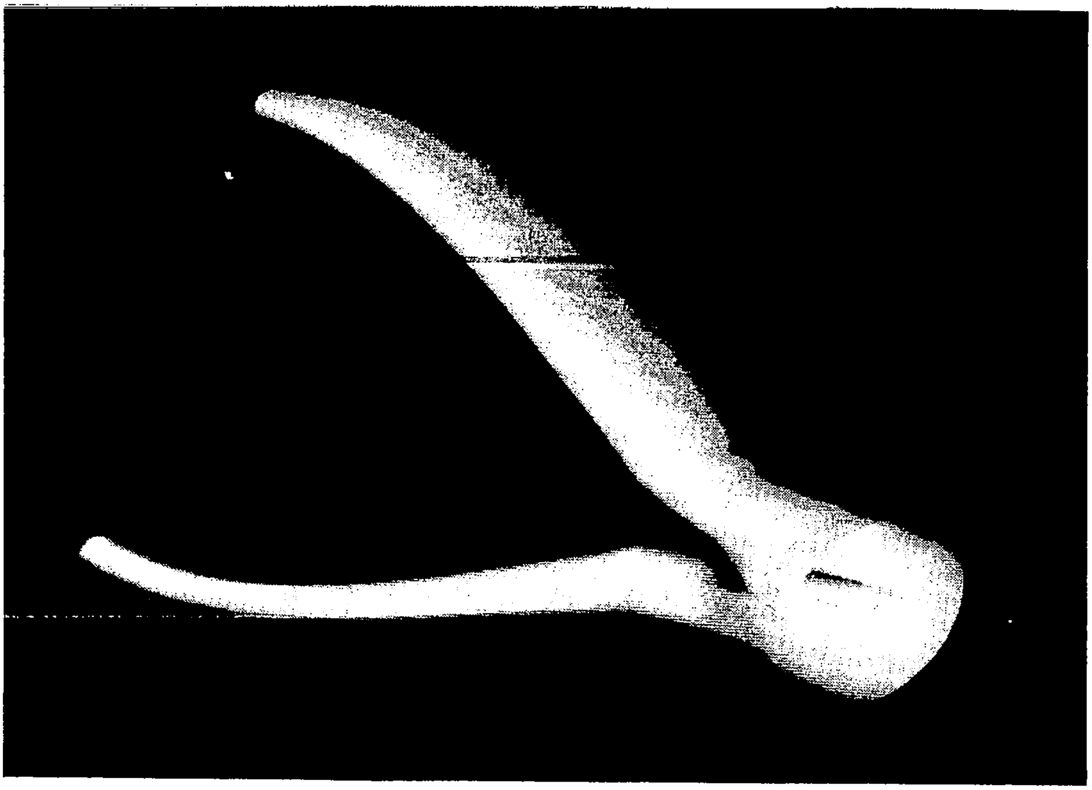
- CYLINDRICAL-COORDINATE PATCH COMPONENTS
- CONSTANT-RADIUS INTERSECTIONS ARE COMPUTATIONALLY PLANAR
- CALCULATIONS ARE UNCHANGED, EXCEPT FOR DISTANCE FORMULAS

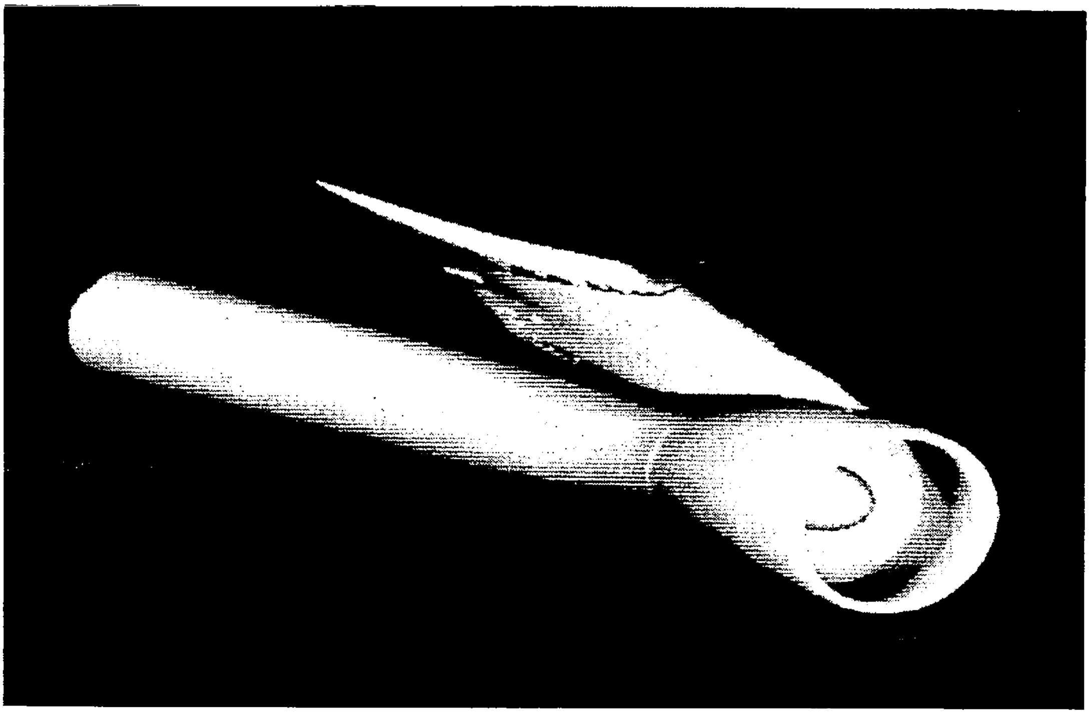
MASTER HAS BEEN TESTED AND APPLIED TO SUPPORT FLOW ANALYSIS

DEFAULT MODELING DETAILS DO NOT DISTURB FLOW ANALYSIS RESULTS

- ANALYTIC SHAPE TESTS: SURFACE DEFINITIONS REPRODUCED FROM MESH CONFIGURATIONS
- OVER 20 INLET CONFIGURATIONS PREPARED IN LAST 3 YEARS
- BARE INLETS ROUTINELY PREPARED WITHIN A WEEK
- SPECIAL CONFIGURATIONS PREPARED: MIXERS AND TURBOPROP INLETS (WITH DUCTS)
- INTERSECTING-OBJECT CONFIGURATIONS PREPARED: SURFACE MODELS TRIMMED







MASTER IS FLEXIBLE AND ACCURATE ENOUGH FOR ENGINEERING USE

IT ADEQUATELY SUPPORTS 3-D FLOW ANALYSIS

- SURFACE-MODELING DETAILS ARE AUTOMATICALLY DETERMINED
- GRAPHIC DISPLAYS CHECK SURFACE MODELS
- CYLINDRICAL COORDINATES ARE SUPPORTED
- INTERSECTING COMPONENTS CAN BE HANDLED
- **MASTER** IS PROVEN BY EXPERIENCE AND REPRODUCTION OF ANALYTIC CONFIGURATIONS

EVALUATION OF 3-D GRAPHICS SOFTWARE
A CASE STUDY

M. E. Lores, S. H. Chasen, and J. M. Garner
Lockheed-Georgia Company
Marietta, Georgia

INTRODUCTION

The primary purpose of an engineering staff of an aircraft company is to produce successful designs that can be built economically. The principal output of the engineering process is thus geometry and the associated manufacturing, assembly, and maintenance specifications. In the past, engineering was done in a series of more or less independent activities. Both the geometric information required by engineering disciplines for input into analyses and the geometry resulting from design studies were transmitted between disciplines by drawings. As a result, the same geometry was re-defined and re-digitized many times--an obviously wasteful and error-prone process. This inefficient process was used because we had no other way to communicate geometric information.

Modern digital computers with their extensive and relatively economical computation and storage capabilities, together with modestly priced high quality interactive graphics devices, offer a means to significantly improve the efficiency with which geometric data are created, manipulated, and maintained. They also allow all engineering disciplines to operate from the same geometric data base. Recently, a number of three-dimensional graphics software systems suitable for geometry applications have become commercially available. Even for companies like Lockheed-Georgia which possess the skills and resources needed to develop 3-D geometry graphics software, the acquisition of commercially available software is an attractive alternative.

Commercial software can be acquired, usually through a lease, for a fraction of the cost needed to develop comparable software. Furthermore, commercial software is maintained, supported, and well-documented by the vendor. And, perhaps most importantly, it is available sooner, resulting in significant indirect cost savings through improved productivity of engineers. In this paper we will briefly discuss our evaluation of 3-D geometry graphics software for possible inclusion in an integrated computer-aided engineering process being implemented at Lockheed-Georgia.

THE EVALUATION PROCESS

Once the need for a computer-aided capability has been established and feasible software has been identified, the first step in selecting a software system is a self-appraisal of current capabilities and computer systems. While the ultimate impact of new computer-aided engineering capabilities may be a revolutionary change in the way engineering is done, the changes must be evolutionary. New systems must co-exist with existing capabilities on which significant time and money have been expended. At the same time, they must provide the basis for desired growth.

At Lockheed-Georgia, we have developed a very efficient 3-D geometry graphics software package which is suitable for advanced design studies. Our advanced design system is called GRADE--Graphics for Advanced Design. Efficiency and ease of use are gained by sacrificing flexibility in surface representation. Our immediate options were either to continue development of GRADE or to acquire a commercially available system which would replace or complement GRADE.

Having received engineering management approval to evaluate candidate software systems and to acquire a suitable system, an ad hoc task force was assembled to develop top-level requirements. The task force was composed primarily of department managers from engineering and manufacturing who have a good understanding of the engineering process and the engineering-to-manufacturing interface. The task force developed and published top-level requirements and then disbanded. These requirements are listed in the charts that follow.

Next, a user evaluation team was assembled and chartered. The team members were from engineering functional organizations such as aerodynamics, structures, advanced design, loft, and from manufacturing. Also, a scientific programmer who has been involved with GRADE from its inception was on the team. We believed it imperative that the evaluation team be made up of users and not members of the company's computer staff.

The evaluation team was rather large, consisting of 18 members. A few users, because of their interest and skill, became very proficient with the software packages and quickly became proxies for other team members. Nevertheless, the spectrum of needs of all potential users continued to drive the evaluation.

The team selected three software packages for evaluation. All three systems met the hard requirements imposed by the ad hoc task force; namely, that they be compatible with CADAM and operate on hardware already in place at Lockheed-Georgia. The costs of the systems are more or less equivalent; hence, cost was not explicitly considered in the evaluation.

The team was trained to use all three systems. They next developed and documented test cases which would reveal the ability of each system to satisfy the requirements. Some of these test cases are shown in the attached charts. The team also developed a scoring method which adequately captured the relative capabilities of the three systems. While more complex multi-attribute decision methods could have been used, we believe the selected method provides all the needed information without being so complex that it is difficult to understand.

An example rating form and the composite scores are included in the attached charts. Notice that even if the value factors are modestly perturbed, system Z is a clear winner based on its overall capabilities relative to our needs. Note also that system Z is superior in two of our vital areas--surfacing and ease of interface with application programs.

CONCLUSIONS

Our lessons learned are listed in the second-to-last chart. A key point arises from the well-defined boundaries that exist between functional organizations in large companies. In this environment, far too much effort can be spent trying to gain agreement between all parties concerning requirements and evaluation results. Indeed unanimity may be impossible to obtain. We must recognize the possible requirement for an authoritative decree directing all affected organizations to implement a selected system. Alternatively, gradual adoption of a new system may result as more and more users become familiar with the system and realize that it makes their work easier and more enjoyable.

As a final observation, we note that modern computers and methods make possible multi-discipline design and optimization studies that offer a means to substantially improve both the quality of our products and the efficiency with which they are developed. However, because this capability breaks down the barriers we have created between functional organizations, we may be witnessing a renewed industry requirement for engineers skilled in multidiscipline design rather than in specialized analysis.

Purpose

Review Our Experiences Acquiring
3-D Graphics Capabilities and Derive
Some Guidelines of General Applicability

Organization

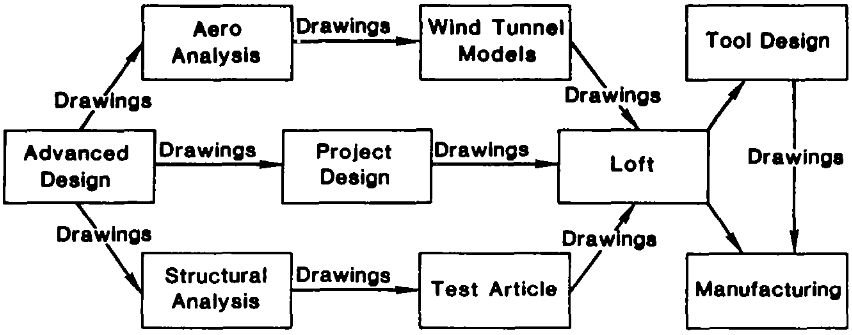
- The Need
- Software Evaluation
- Implications

The Need:

The Primary Purpose of Engineering Is To Produce Successful Designs Which Can Be Built Economically

Geometry Is the Principal Output of Engineering

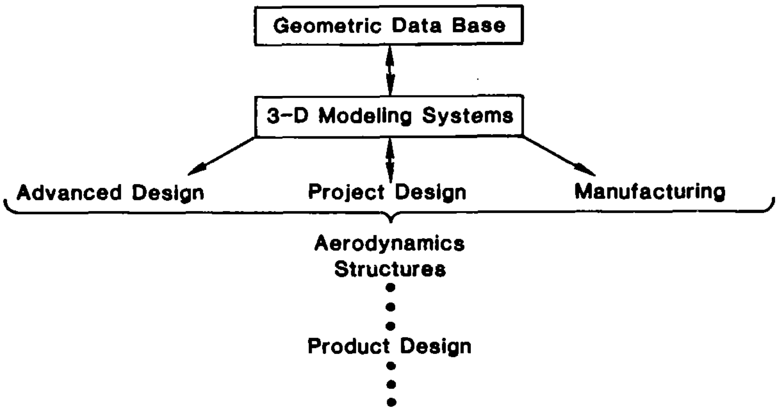
Current Design Process



- Repetitive
- Error Prone

Done Because We Had No Other Way To Convey Geometric Data

Design With 3-D Modeling



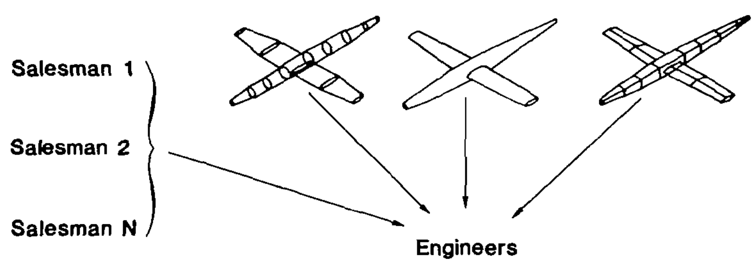
Why Purchase Software?

Cost, Time and Demonstrated Capability

But:

- Must Satisfy a Well-Defined Need
- Must Be Well Supported

The Evaluation



Can You Really Do That?

Lockheed-Georgia Status in 1981

- Extensive Analysis and Design Capability in Functional Organizations
- Extensive Use of 2-D CADAM
- 3-D Surface Modeling System Suitable for Conceptual Design
- Heterogeneous, Disconnected Computer System
- Newly Formed Department To Interface Engineering Computer Operations

The Immediate Decision

- Continue Development of In-House 3-D Surface Modeling System
- Acquire a Commercially Available System Which Either Replaces or Complements In-House System

Steps in Our Evaluation

1. Initial Exposure to an Existing System at a Conference
2. Management Support To Evaluate Systems
3. Ad hoc Task Force To Develop Top-Level Requirements
4. User Evaluation Team Assembled
5. Candidate Systems Selected
6. Test Cases and Evaluation Criteria Selected
7. Training and Evaluation
8. Recommendation to Management

Essential Requirements

1. Mathematically Describe Any Surface With Loft-Quality Models
2. Provide Automatic Sectioning and Intersections
3. Perform Volume Calculations
4. Provide Kinematics Functions
5. Provide Numerical Control Capabilities Compatible With APT Equivalent Postprocessors
6. Operate on Existing or Projected Computer Hardware

Essential Requirements (Con't)

7. Interface With CADAM
8. Provide Practical Interface With Applications
9. Provide Practical Interface With Existing Peripherals Such as Plotters
10. Provide Backup and Data Recovery Techniques Similar to CADAM
11. Be Well Documented
12. Be Supported by Reputable Vendor

Desirable Features

1. Support Rapid Initial Configuration Definition and Easy Modifications
2. Interface With Finite Element Modeling Already Developed and F.E.M. Available
3. Support Robotics
4. Provide 3–D Inspection Data Compatible With APT Equivalent Postprocessors
5. Be Modular in Design

Desirable Features (Con't)

6. Provide Hidden Line Removal
7. Provide Shading
8. Support Color Hardware
9. Support Touch–Screen Hardware
10. Support Bit–Pad Hardware, Light Pen Emulation, and Digitizing

User Evaluation Team

- Not Necessarily the Same as the ad hoc Task Force
- Attempted To Represent Spectrum of Users
  - Functional Engineering
  - Advanced Design
  - Loft
  - Manufacturing

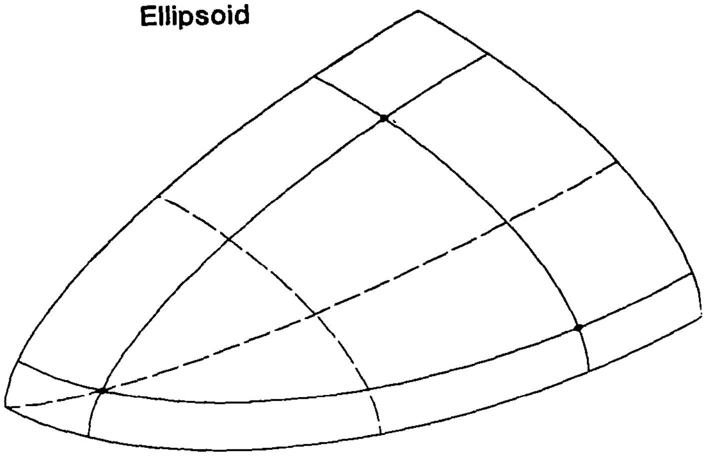
  $\left.\right\} \longleftrightarrow$ Scientific Programming Support
- Impartial Chairman from Scientific Programming

Training and Evaluation

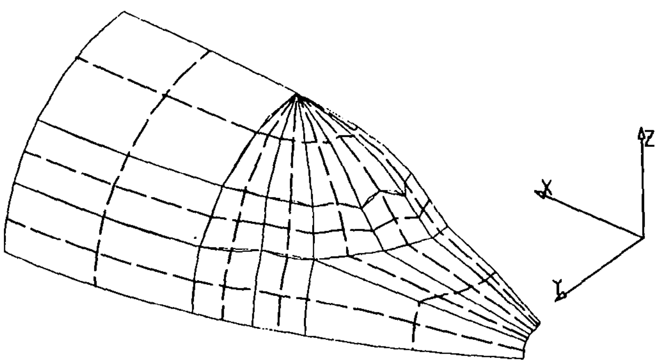
- 130 Hours Training per Team Member
- 400 to 1000 Hours Evaluation per Team Member
- One Year Evaluation

Accuracy:

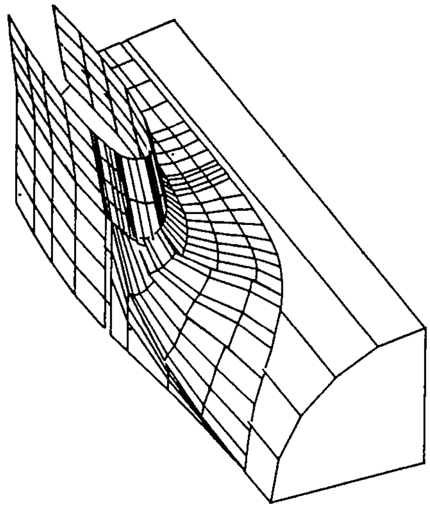
Ellipsoid



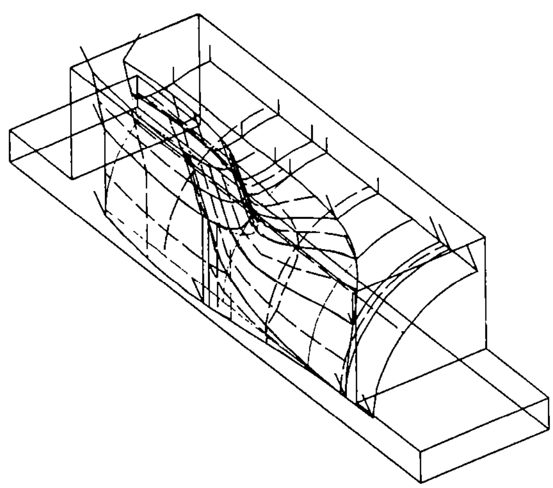
Complex Surface - CAB



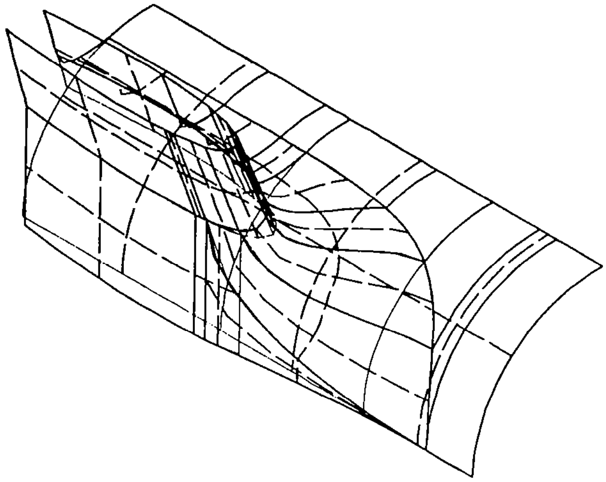
Complex Surface – Fillet



Complex Surface – Machining

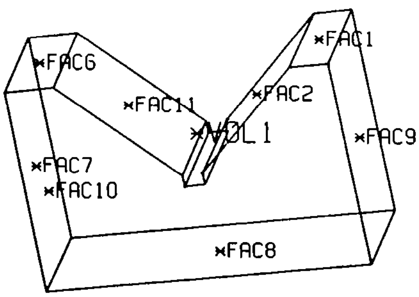


Complex Surface - Design



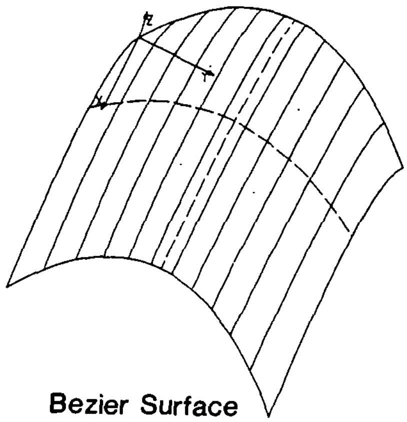
Versatility:

Volume Calculation



Flexibility:

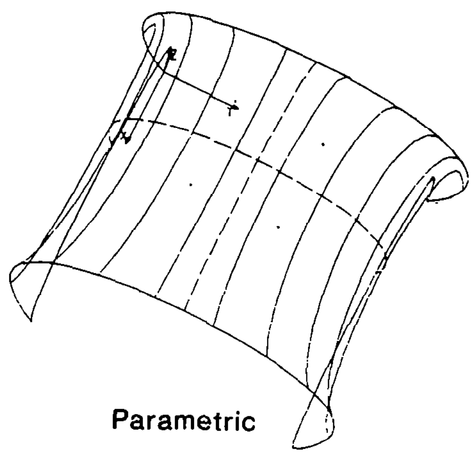
Surface Through Points



Bezier Surface

Flexibility:

Surface Through Points



Parametric

Scoring Procedure

1. Fourteen Test Cases To Evaluate Different Features
2. Evaluators Assigned to Test Cases
3. For Each Test Case, Ability To Perform Certain Functions Scored from 1 to 5
4. Average Score Computed for Test Case
5. Average Scores Weighted and Composite Score Computed

Example Test Case

Test Case No. 1

Test Case Name: Fuselage, Wing, and Wing to Fuselage Fillet Using Data from GRADE

Evaluator: R. Smith

Primary Purpose of Test: Feasibility of Using GRADE Data To Generate Aircraft Configuration

Description: Fuselage and Fillet Data from GRADE and Wing Data from Aerodynamics. Windshield Planes Will Be Determined by the Evaluator and Will Not Necessarily Meet Appropriate Specifications. The Surface Tangents Will Be Controlled Where Required for Surface Continuity

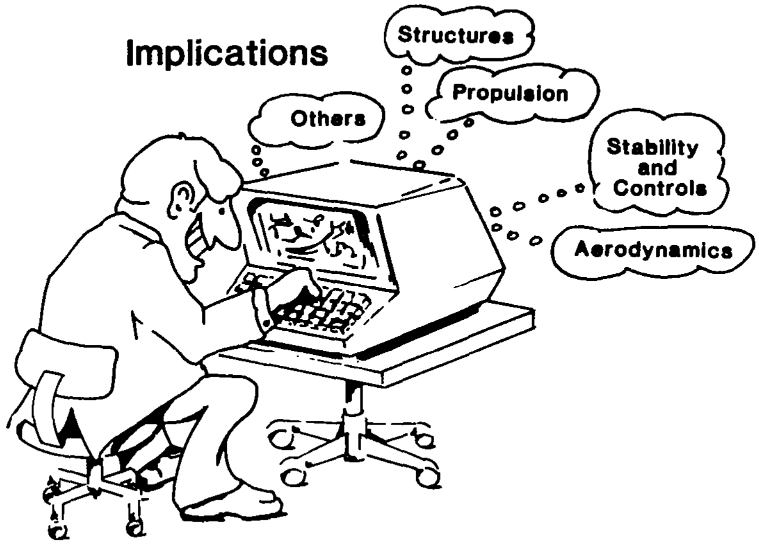
Example Test Case

Test Case No. 1	System		
Functions	X	Y	Z
1. Conics from GRADE	4	3	3
2. Cubics from GRADE	2	2	2
3. Quartics from GRADE	2	2	2
4. Ease of Construction from GRADE Data	4	2	2
5. Fairness of Surface	2	1.5	4
6. Accuracy (Compared With GRADE's Surfaces)	3	3	4
7. Cross Boundary Slope Match	2	2	4
8. Adjustment Capability	2	1	4
Average Scores	2.63	2.06	3.13

Composite Scores

Capability	Weight Factor	System		
		X	Y	Z
1. Surfacing	10	21	24	31
2. Interfacing	10	10	35	45
3. NC	7	0	0	21
4. Soft Mock Up	6	19	5	17
5. Kinematic	5	11	16	16
6. Construction of 3-D Parts	4	10	5	0
7. F.E.M.	3	0	8	13
8. Construction of 3-D Parts as Solids	3	0	0	7
Total Evaluation Scores	47	71	93	150

Implications



Lessons Learned:

1. An Instigator Is Needed

2. Existing Hardware, Software, and Procedures Must Be Considered in Requirements: Changes Will Be Evolutionary

3. Unanimity in Requirements and Evaluation Is a Worthy Goal, but Not Possible

4. Rely on People Who Show an Interest

5. Training and Learning Take Time

6. At Some Point, A Decree Will Have To Be Issued

Issues:

- Computers Make Possible Multi-Discipline Design and Optimization
- Traditional Post-WW II Functional Organization May Not Be Appropriate
- Emphases Should Be Placed on:
  - Data Base Management Systems
  - Inter-Computer Communications
- Universities Should Reemphasize Aircraft Design

USING A COMMERCIAL CAD SYSTEM FOR SIMULTANEOUS INPUT TO THEORETICAL AERODYNAMIC PROGRAMS AND WIND-TUNNEL MODEL CONSTRUCTION

Francis Enomoto and Paul Keller

NASA Ames Research Center, Moffett Field, California

INTRODUCTION

Engineers in the Aircraft Aerodynamics Branch at Ames Research Center are using a commercial computer-aided design (CAD) system as a tool to improve the efficiency of aerodynamic research (fig. 2). The system uses a common geometry database to generate input for theoretical programs and input for numerically controlled (NC) machine tools for fabricating wind-tunnel parts. Although the CAD system provides a general capability for constructing geometry, finite-element models, and NC tool paths, it still had to be adapted to effectively aid in aerodynamic research. Software was developed to assist in constructing complex geometries and to generate inputs for theoretical programs. This paper describes the approach and software used to construct geometry for aircraft configurations, to generate the input for a panel-method analysis program, and to define NC tool paths.

SYSTEM DESCRIPTION

The commercial CAD system is the Calma Design, Drafting, and Manufacturing (DDM) System. It is a minicomputer based interactive graphics system with full 3-dimensional capabilities. The user interacts with the system using the DDM graphics command language at a graphics workstation shown in figure 3. An important feature of DDM, which allows the user to customize the system for his needs, is the Design Analysis Language (DAL). DAL combines all of the DDM graphics commands with arithmetic, logic, looping, text manipulation, and file handling commands.

GEOMETRY CONSTRUCTION

Starting with sketches, engineering drawings, or lists of coordinates, the user constructs a 3-dimensional model of an aircraft configuration. Figure 4 shows the three steps to construct a surface. Points are first constructed by typing in coordinates. These points are used in DDM commands to construct lines, arcs, conics, or B-splines. For example, a B-spline is created from points selected in the desired order with optionally specified tangent vectors at the end points. Other DDM commands are then used to construct several types of surfaces from this geometry. The types normally used are: ruled surfaces for wings, surfaces of revolution for fuselages, B-surfaces for nacelles and fairings. These surfaces are the main basis from which to generate input for theoretical programs and NC tool paths. If geometry information is stored in computer files, such as wing section coordinates, DAL programs are used to read in this information and to automatically construct points, splines, or surfaces.

Extensive DAL software is being developed to automate the modeling of aircraft configurations (fig. 5). The user will be able to construct wings, tails, fuselages, and nacelles by specifying basic geometric parameters.

THEORETICAL ANALYSIS INPUT GENERATION

The geometry input used by the panel-method analysis program, PAN AIR, is a set of nonplanar rectangular meshes of points called networks. These networks define the panels which represent the aircraft surface. DAL programs and DDM commands can generate grid points for the networks by two methods, then write their coordinates to a file in the proper PAN AIR format.

One method for creating networks is with a DAL program that generates grid points on a surface defined in the geometry database (fig. 6). This method is typically used for wing and fuselage surfaces. The user selects the surface and then specifies the desired number of rows and columns of points. The type of point spacing in each direction can be: equal, cosine, sine, double cosine, and user defined. The program translates these parameters into a set of coefficients that are used to compute the location of any point in the surface parametric space, F(u,v). The user assigns a name to the group of grid points after they are generated. This name is used to access the points when writing their coordinates to a PAN AIR input file.

More control over the placement of grid points on a surface is obtained with a second method that uses splines generated on the defined surfaces (fig. 7). This method is used when aligning the distribution of grid points of one network with that of previously defined networks, such as at the nacelle-wing intersection. These splines are generated with a DDM command that computes the intersection of a user specified plane with a surface or at the intersection of two surfaces. Grid points are then generated on each spline with a DAL program similar to the one used for surfaces by specifying the number of points and the type of spacing. If sets of longitudinal and cross splines are generated on the surface, then the intersections of the two sets of curves are used as the network grid points.

A DAL program creates a PAN AIR input file. The user specifies the file name, the name of the group of grid points, the PAN AIR network name, the number of rows and columns of points, and the PAN AIR aerodynamic boundary condition. The program then opens the file on the system disk and writes out the coordinates along with the network and boundary condition lines. The user has the option to write out several networks or to use the program several times to concatenate each network into a single file.

WIND-TUNNEL MODEL PARTS FABRICATION

Wind-tunnel model parts are fabricated from NC tool paths generated on the CAD system. The tool paths are generated using the Calma supplied NC applications package (fig. 8). This software can generate 2-, 3-, and 5-axis tool paths. The user specifies parameters, selects the surfaces or boundaries, and indicates the initial direction and appropriate side to be cut. The tool path is computed and then displayed on the screen. If the user is satisfied with the tool path, it is then written to an Automatically Programmed Tools (APT) source file. The APT file is post-processed on another computer system to produce the punch tape that controls the milling machine.

EXAMPLE OF PROCESS: ADVANCED TURBOPROP AIRCRAFT MODEL

The first production project on the CAD system was to construct the geometry database for an existing advanced turboprop semi-span wind-tunnel model (fig. 9). This database was used to create the PAN AIR analysis input of the model using the grid point generation methods described earlier (fig. 10). NC tool paths were also generated to machine a contoured wooden nacelle (fig. 11). This part was used as a mold in the fabrication process of fiberglass shells for the wind-tunnel model.

CONCLUSION

The CAD system's common geometry database was used to generate input for theoretical programs and NC tool paths for wind-tunnel part fabrication. This eliminates the duplication of work in generating separate geometry databases for each type of analysis. Another advantage is that it reduces the uncertainty due to geometric differences when comparing theoretical aerodynamic data with wind-tunnel data.

The system was adapted to aerodynamic research by developing programs written in DAL. These programs reduced the amount of time required to construct complex geometries and to generate input for theoretical programs. Certain shortcomings of the DDM software limited the effectiveness of these programs and some of the Calma NC software. The complexity of aircraft configurations suggeststhat more types of surface and curve geometry should be added to the system. Some of these shortcomings may be eliminated as improved versions of DDM are made available.

OBJECTIVES

- SYSTEM DESCRIPTION
- GEOMETRY CONSTRUCTION METHODS
- THEORETICAL ANALYSIS INPUT GENERATION METHODS
- NC TOOL PATH GENERATION
- EXAMPLE OF PROCESS

FIG 1

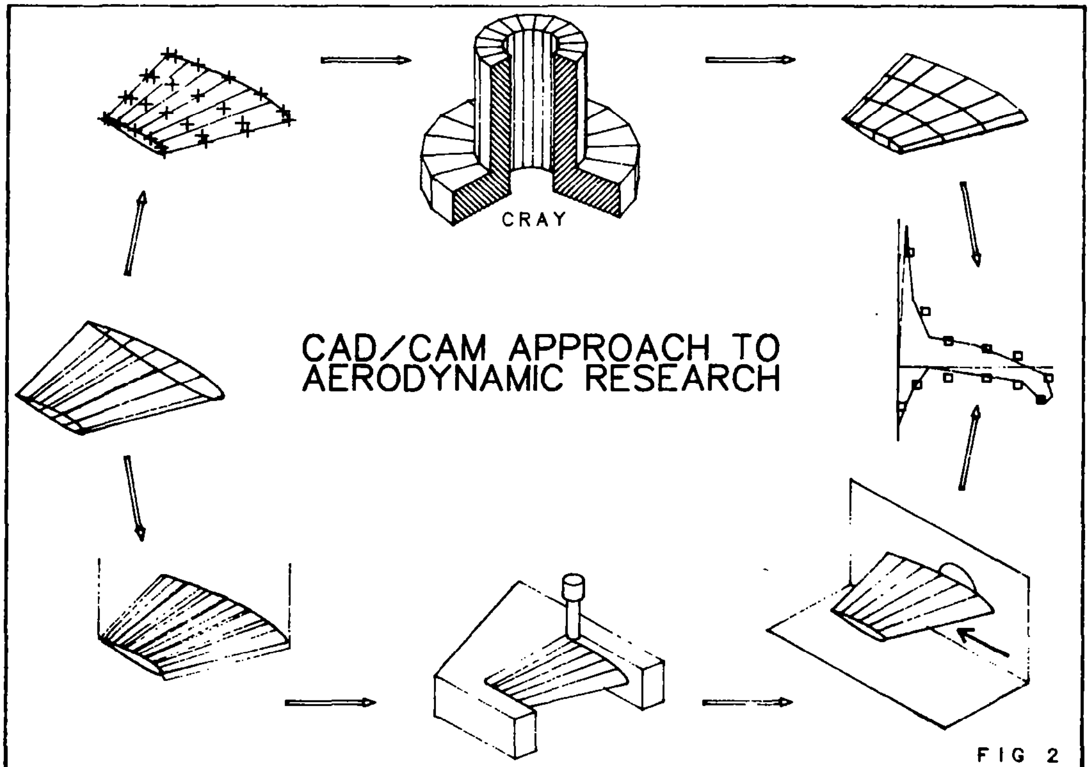


FIG 2

CALMA GRAPHICS WORKSTATION

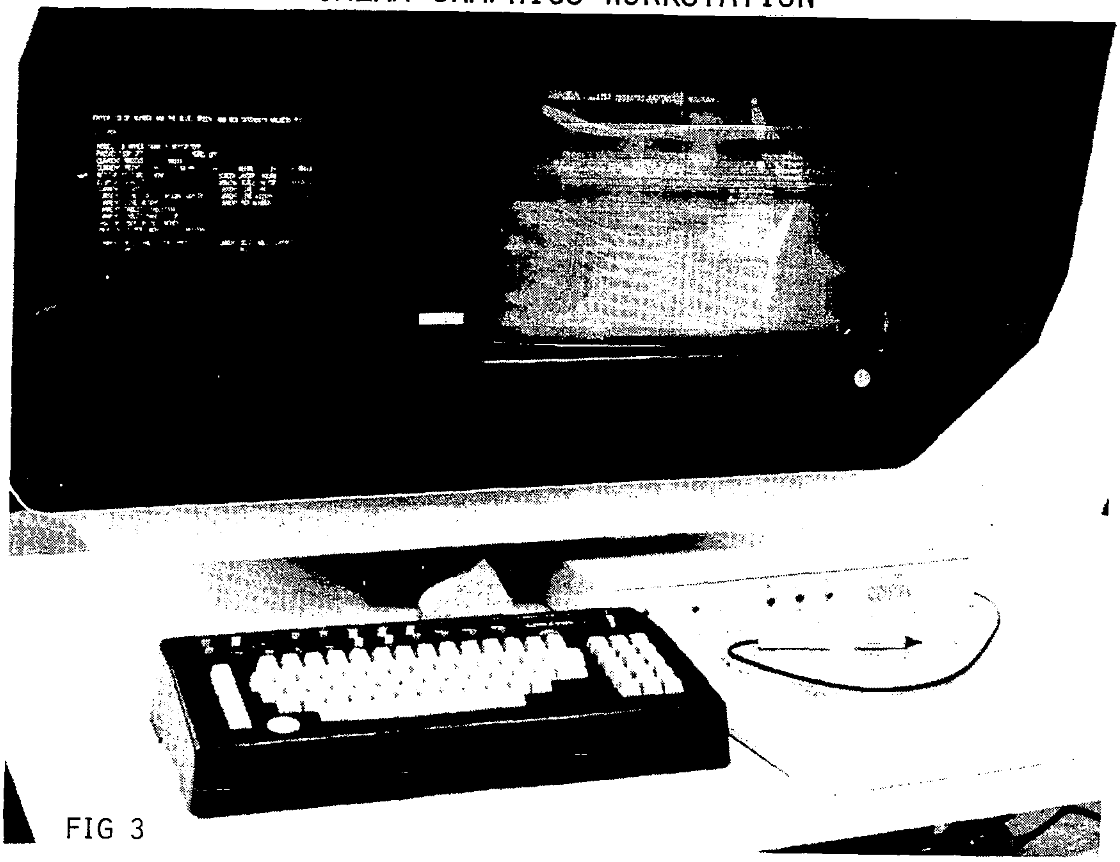


FIG 3

SURFACE CONSTRUCTION

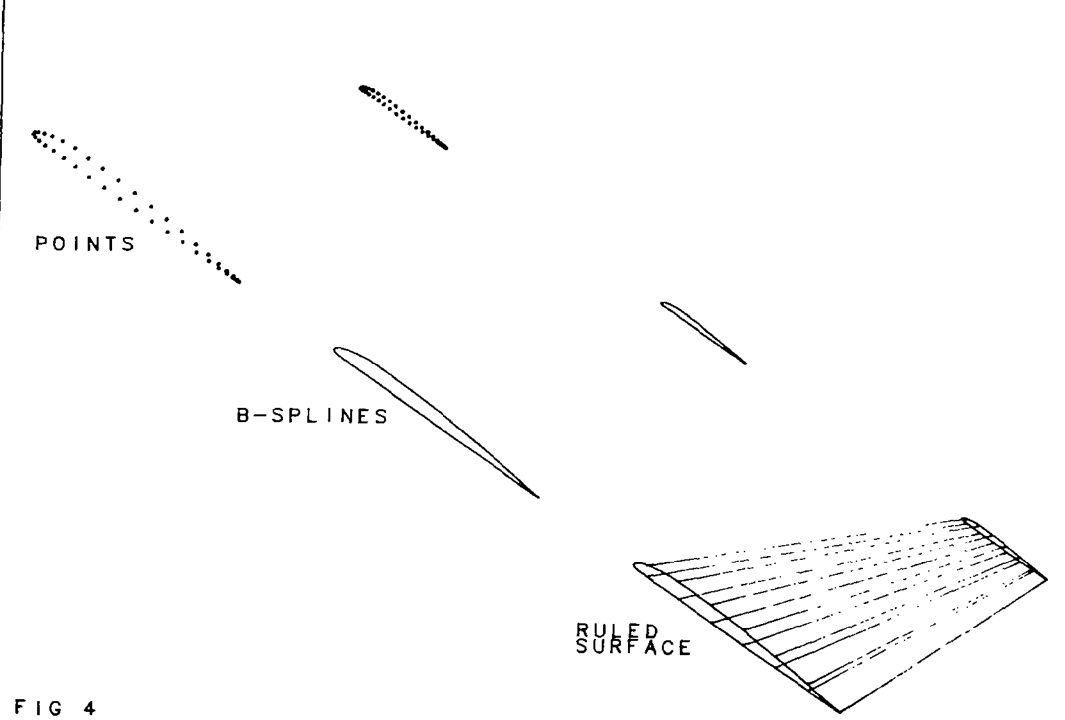


FIG 4

AUTOMATED CONSTRUCTION OF AIRCRAFT GEOMETRY

THESE ARE THE CURRENT WING (FORWARD WING) PARAMETERS:

1.	SEMISPAN:	24.809 IN	LOCATION OF ROOT L. E.:	
2.	ASPECT RATIO:	2.571	7.	X= 39.30 IN
3.	TAPER RATIO:	.413	8.	Y= 0.00 IN
4.	L. E. SWEEP ANGLE:	36.072 DEG	9.	Z= -3.95 IN
5.	DIHEDRAL ANGLE:	5.000 DEG		
6.	AIRFOIL (NACA 4 OR 5 DIGIT OR T/C):	4210		

SELECT THE OPERATION YOU WISH TO PERFORM:

1. EDIT PARAMETER LIST
2. CREATE MODEL
3. STORE DATA, EXIT
4. DO NOT STORE DATA, EXIT
5. CALCULATE EXPRESSION
6. DELETE LAST ITEM
7. MAG ALL, REPAINT
8. MISC. OPERATIONS

FIG 5A

AUTOMATED CONSTRUCTION OF AIRCRAFT GEOMETRY

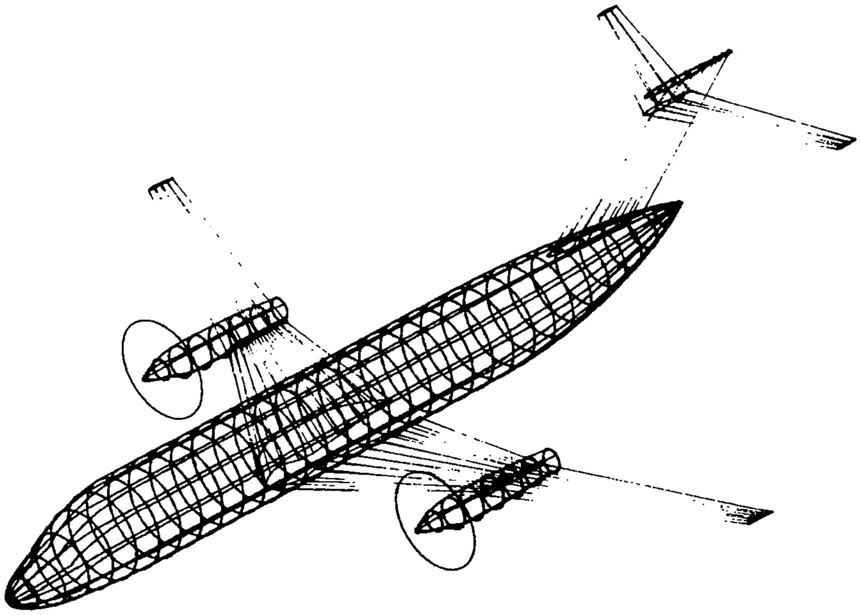
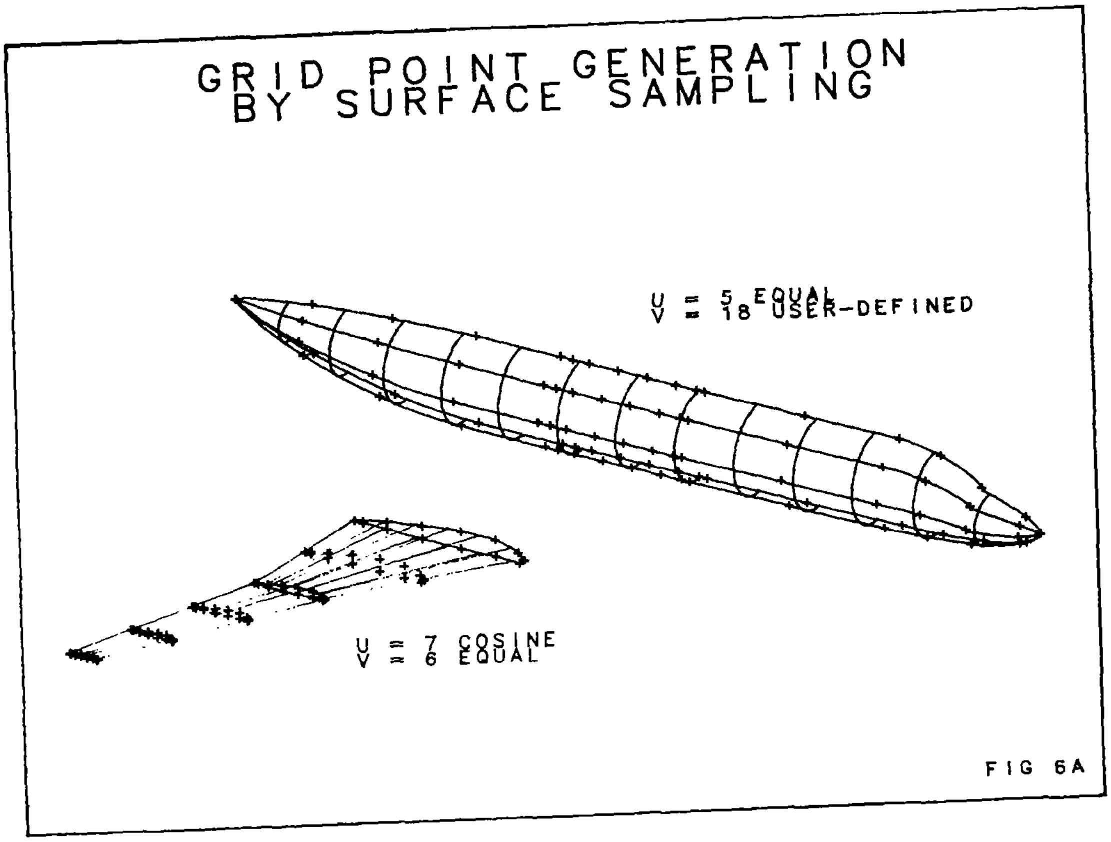
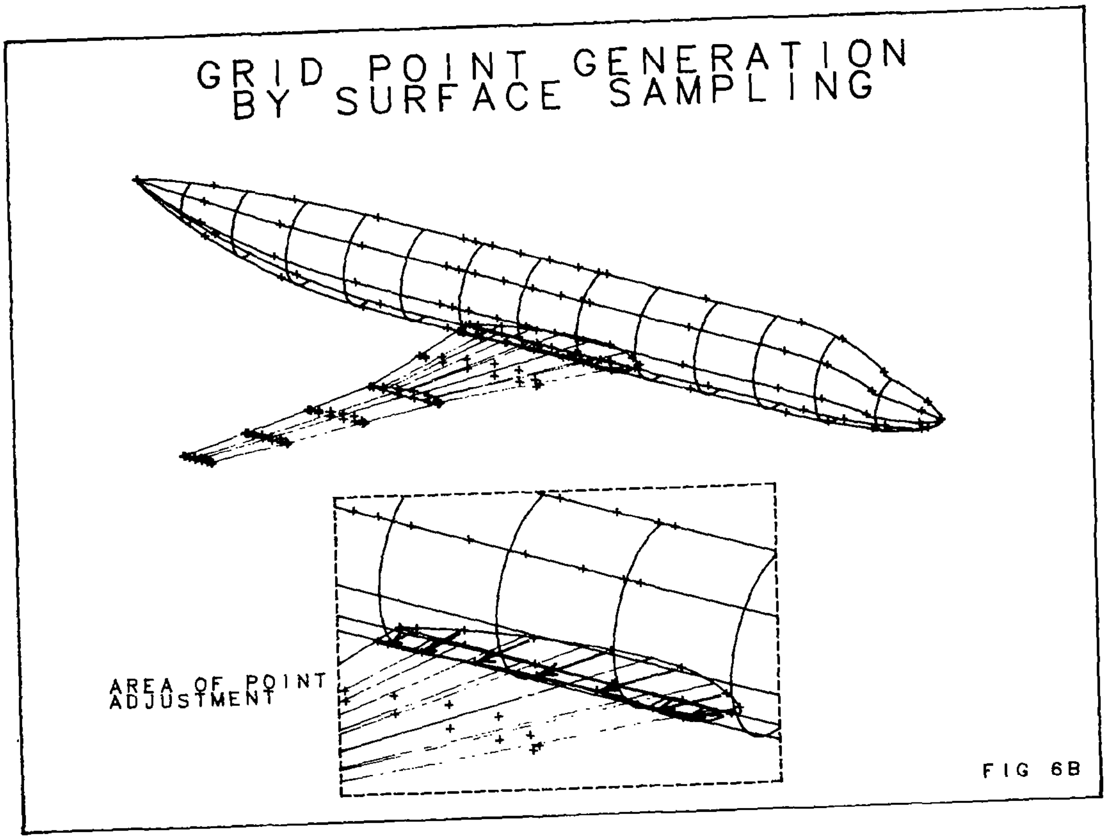


FIG 5B


GRID POINT GENERATION
BY SURFACE SAMPLING
U = 5 EQUAL
V = 18 USER-DEFINED
U = 7 COSINE
V = 6 EQUAL
FIG 6A


GRID POINT GENERATION
BY SURFACE SAMPLING
AREA OF POINT
ADJUSTMENT
FIG 6B

GRID POINT GENERATION USING CROSS-SECTION SPLINES

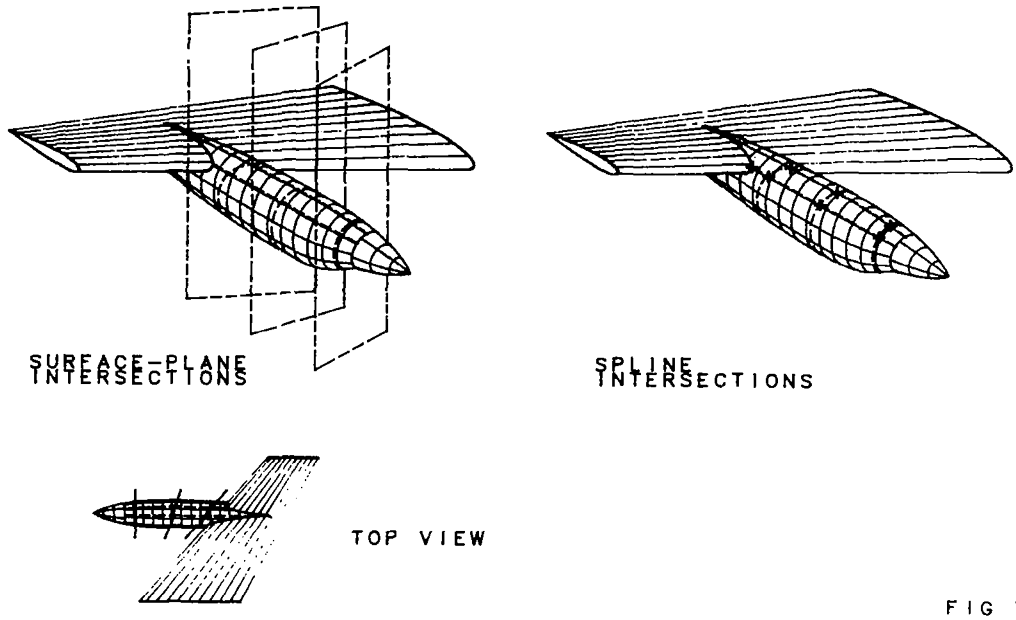


FIG 7

NC TOOL PATH GENERATION

3-AXIS TOOL PATH
GENERATION MENU

1. VERIFY 3-AXIS PARAMETERS
2. TOOL NAME
3. CLEARANCE/RETRACT PLANES
4. APPROACH/RETRACT
5. LACE CUTTING
6. NUMBER OF SURFACE CUTS
7. MINIMUM/MAXIMUM STEP SIZE
8. THICK
9. FINISH TOLERANCE
10. NORMAL ROUGH CUTS
11. ROUGH TOLERANCE
12. TYPE OF CONTAINMENT
13. CONTAINMENT BOUNDARY SAMPLING TOLERANCE

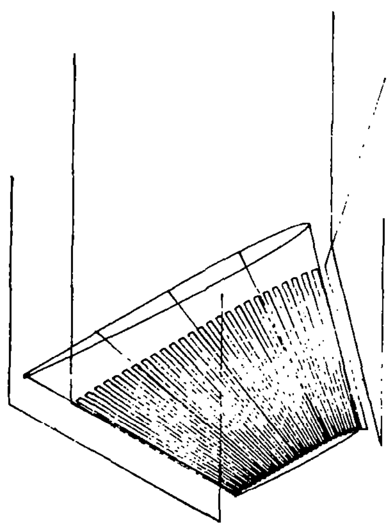
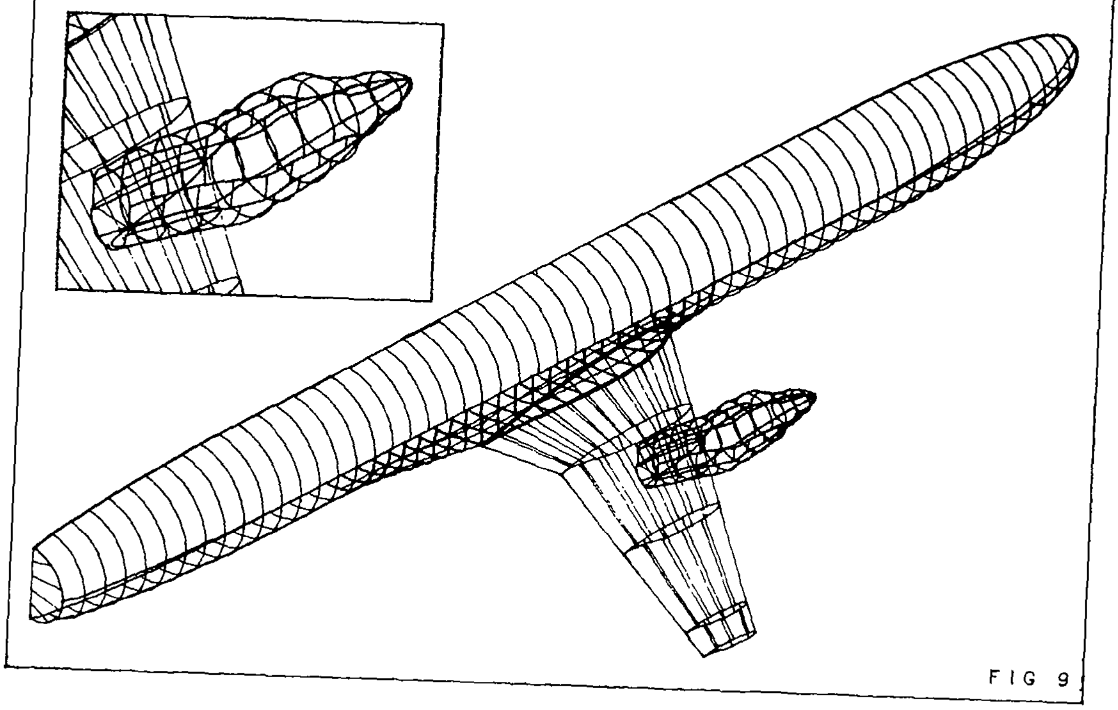
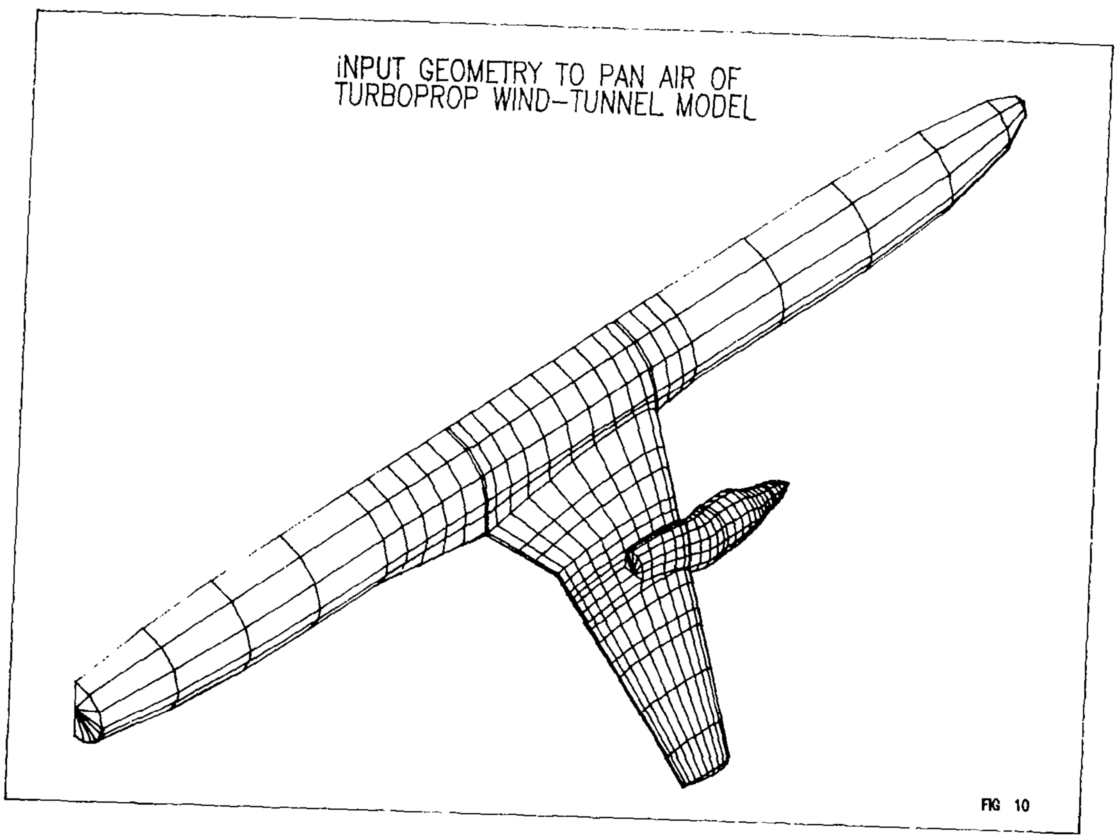


FIG 8



CAD GEOMETRY DEFINITION OF TURBOPROP WIND-TUNNEL MODEL

FIG 9



INPUT GEOMETRY TO PAN AIR OF TURBOPROP WIND-TUNNEL MODEL

FIG 10

NC TOOL PATHS ON CONTOURED NACELLE

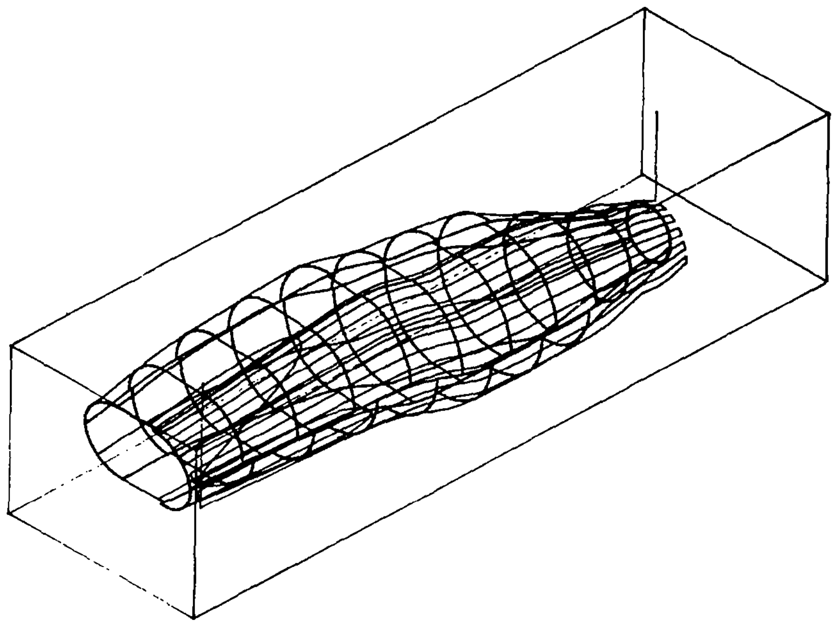


FIG 11

CONCLUSIONS

- COMMON GEOMETRY DATABASE ADVANTAGES

- SYSTEM ADAPTATION THROUGH DAL

FIG 12

MATHEMATICAL SYNTHESIZATION OF COMPLEX STRUCTURES

L. Bernard Garrett
NASA Langley Research Center
Hampton, Virginia

INTRODUCTION

This paper describes a mathematical synthesization approach for rapidly modeling and analyzing complex structures comprised of hundreds or thousands of repeating individual structural members, interconnecting hardware, and other components. The techniques are applicable to any structure with repeating structural members; however, the interactive computer programs discussed herein are specifically tailored to future large spacecraft such as the large antenna spacecraft concepts depicted in figure 1. The capabilities are embodied within the Interactive Design and Evaluation of Advanced Spacecraft (IDEAS) computer-aided design system. IDEAS has a full range of integrated spacecraft modeling, design, analysis, performance, and cost estimating capabilities which include about 40 technical program (refs. 1-7) and executive, data base (ref. 8), and file management systems as shown in figure 2. With IDEAS, a single user at an interactive terminal can create, design, analyze, and conduct parametric studies of Earth orbiting spacecraft in a timely, cost-efficient manner. The system is particularly useful in the conceptual design phases of advanced space missions when a multiplicity of concepts must be evaluated.

The IDEAS interactive finite-element modeling capabilities are discussed in this paper. The mathematical synthesization programs have been structured not only to provide a detailed finite-element model of the structure but also to automatically perform the many auxiliary computations needed to analyze the spacecraft in its orbital operational environment.

The approach in IDEAS is to use strict mathematical relations to automatically generate a finite-element model of the repeating structural members of the dish (for any user-specified size, shape, and lattice density). Then, the interconnecting dish hardware masses and sizes are computed as functions of the structural member diameter and their masses distributed at appropriate nodal points in the dish finite-element model. Additional structural appendages are then added to the dish using the appendage synthesizer programs. Finally, spacecraft subsystems are placed at user-specified node points to complete the initial definition of the total spacecraft finite-element model in a form suitable for subsequent static loading, thermal, and dynamic analyses. Spacecraft mass, inertia, drag and solar pressure areas, and centers of gravity and pressure are automatically computed for subsequent rigid body controls analyses and subsystem sizing. Other programs such as surface accuracy distortion, radio frequency analysis, and costs access the data base and files created by the foregoing programs to generate performance and cost estimates.

DISH STRUCTURE SYNTHESIZERS

Five mathematical synthesizers are available in IDEAS to rapidly create finite-element models of a dish and/or platform as shown in figure 1. Consider, for example, the Tetrahedral Truss Structure Synthesizer (TTSS) module. This module rapidly models a flat or curved (parabolic or spherical) tetrahedral truss structure and initially sizes the structural members. The module automatically generates the nodal geometry; the member connectivity, cross-sectional areas, and masses; and the resultant finite-element model of the structure for a specified dish diameter, shape, number of bays, and truss depth. The tetrahedral truss configuration definition and major hardware components are shown in figure 3(a).

The truss structural members can be circular tubes, isogrids, or trianglar-truss struts as shown in figure 3(b). The surface and diagonal members are sized separately for Euler buckling from input material properties and initial loading conditions. Structural members can be constrained to minimum material thicknesses and tube diameters so that the column buckling equations will not design members too small for practical use. An option also permits the sizing of the member (diameter and thickness) from user-specified length over radius of gyration and tube radius over thickness ratio inputs.

Masses of the folding hinges, spiders, bearings, and end fittings are computed as functions of the structural member diameter. A mesh reflective surface and the support system may be optionally included in the calculations and are automatically distributed at each nodal point on one of the surfaces. The mesh control system, used to maintain contour, is computed as a percentage of the mesh weight. A contingency mass is included which is defined as a percentage of the mass of all the structural components.

Forty-five input variables, as shown in table I, are needed to run TTSS. The module calculates and outputs the masses of the structural components, mesh system and total system, the center of gravity, and mass/inertia properties. The displayed outputs also include structural member dimensions, hardware part counts, unit masses, total group masses, mesh area, and configuration packaging dimensions as shown in table II. Uniquely nameable data base and files, including a complete finite-element model and mass properties of the tetrahedral truss, are created in TTSS for later use in other modules.

In about 5 wall-clock minutes on the minicomputer, TTSS can detail all 6864 struts of a 32-bay tetrahedral dish of any diameter, curvature, and truss depth, size the structural members, incorporate all joints, pins, hinges, and reflective mesh systems, display the output summary data and the structural finite-element model, and write the data-base/file information to retrievable permanent disk storage. An eight-bay case (420 elements) can be completed in 1 to 2 wall-clock minutes on the minicomputer. Similar synthesizer capabilities exist for the other structural concepts.

ANALOGOUS MODELING

The ANALOG module is a preprocessor for TTSS and permits a large number of tetrahedral truss members in a flat platform or curved dish truss to be replaced for analysis by a smaller number of equivalent members, as illustrated in figure 4. The approach and transformation equations, developed by Leondis (ref. 3), are based on the use of a set of Constitutive Relations (stiffness characteristics) for repeating plate-like lattice structures such as a tetrahedral truss. Member sizes and geometry, physical properties, and loads are transformed in such a manner as to retain the equivalent strength, stiffness, inertia, and thermal characteristics of the original structure. The analogous modeling capability is particularly useful for rapid parametric analyses since run times and core sizes are reduced by 50 to 90 percent.

APPENDAGE SYNTHESIZERS

These programs allow the user to design and add structural appendages to the dish or platform and to locate spacecraft subsystems on the structure. The process results in an updated finite-element model and mass/inertia properties for the entire spacecraft. An example of a final configuration for the spacecraft is shown in figure 5 which depicts the dish, added appendages, and subsystems required for an autonomous Earth-orbiting microwave radiometer satellite.

Standard user-selectable structural members include the hollow rod, a mass-efficient isogrid, a relatively stiff triangular strut, and tension cables as shown in figure 3(b). The isogrid and triangular strut members are reduced to equivalent hollow rods which retain the same stiffness, strength, and mass of the original members and are automatically designed to user-specific Euler buckling loads. Intermediate node points may be specified for each structural member to yield a higher fidelity finite-element model for modal analyses.

Another category of appendage synthesizers allows the user to design deployable structural appendages. These classes of deployable appendages are shown in figure 6. They are elastically deformable longeron masts (usually limited to about 1-m diameter), two types of triangular masts (usually limited to less than 3-m diameter based on Shuttle Orbiter packaging volume constraints), and box-truss masts (which are repeating elements of the box-truss dish shown in figure 1 that can range up to 15 to 20 m in width in their deployed state). Masses of the hardware interconnects and deployment systems are also accumulated in the synthesization process.

DRAG AREA COMPUTATIONS

Another labor-saving feature incorporated into the IDEAS system is the automated computation of effective drag areas for lattice structures with porous reflective antenna mesh. The process is generally lengthy and time-consuming. However, it is mandatory that this be an input for computation of orbital propellant requirements. In single discipline analyses, the structural analyst may have no need for these areas and the aerodynamicist is sometimes faced with the formidable task of generating these data for various structural elements and orientations from design drawings. Drag areas can range from a low percent of the solid spacecraft area up to multiples of solid area depending on such factors as the spacecraft size and orientation and the type and quantity of structural members. The IDEAS programs rapidly accumulate these areas from the data base and finite-element model data created in the synthesizer programs.

The atmospheric drag area approximation approach is illustrated in figure 7. Each node, which consists of the intersection of several structural members at various orientations, is reduced from the finite-element model to an equivalent solid structural area normal to the spacecraft velocity vector. The blockage effects of upstream areas on downstream areas are factored into the solution. In addition, the program will incorporate the drag effects of the variable transmissibility mesh into the solution from user-specified inputs on the porosity of the reflective mesh and the orientation angle at which mesh appears solid to the incident molecules. Solid areas of supporting subsystems (such as solar arrays) are included.

FINITE-ELEMENT MODELING OF DEPLOYING STRUCTURES

Limited research effort has been devoted to the important areas of deployment, kinematic, and kinetic analyses of large spacecraft. The spacecraft undergoes large changes in inertia (see figure 8) and center of gravity location that may affect spacecraft stability and influence rigid body control system design. The capability to generate finite-element models of the structures as they undergo deployment is a prerequisite to conducting stability and stress analyses. The application of mathematical synthesization approaches to describe the deploying structure offers a rapid, effective means for generating the needed finite-element models at any stage of deployment.

REFERENCES

1. Garrett, L. Bernard: Interactive Design and Analysis of Future Large Spacecraft Concepts. NASA TP-1937, 1981.

2. Leondis, Alex F.: Large Advanced Space Systems (LASS) Computer Program. AIAA Paper 79-0904, May 1979.

3. Leondis, Alex F.: Large Advanced Space Systems Computer-Aided Design and Analysis Program. NASA CR-159191, 1980.

4. Campbell, R. H.: Spacecraft Design and Cost Model Development. Final Report, Aerospace Corporation Report ATM-76(8191)-3, June 30, 1976. (See also Campbell, B. H.: Systems Cost/Performance Analysis. Aerospace Corporation Reports ATM-75(7363)-3 Vol. I-II, March 1975).

5. Farrell, C. E.; and Zimbelman, H. F.: Advanced Space Systems Analysis Software--Technical, User and Programmer Guide. NASA CR-165798, September 1981.

6. Darby, H. R.; Coomer, T. N.; and Collins, F. M.: Manual for the Space Station Conceptual Design Model, General Dynamics Corporation, Fort Worth, Report MR-0-243, April 1969.

7. Farrell, C. E.: Advanced Earth Observation Spacecraft Computer-Aided Design Software--Technical, User, and Programmer Guide. Final Report NASA CR-166063, December 1982.

8. Wilhite, A. W.; and Rehder, J. J.: AVID: A Design System for Technology Studies of Advanced Transportation Concepts. AIAA Paper 79-0872, May 1979.

Table I. TTSS input variables.

55 000	1	RFDM	-RADIO FREQUENCY DIAMETER (METERS)
1 0000	2	SHAPE	-SHAPE FLAG 1-PARABOLA, 2-SPHERE, 3-FLAT
1 5000	3	FOVERD	-FOCAL LENGTH TO RF DIAMETER RATIO
8 0000	4	NBAYS	-NUMBER OF BAYS IN REAL DISH STRUCTURE
8 0000	5	ANBAYS	-ANALYSIS NUMBER OF BAYS
0 30000	6	THETA	-DIAGONAL ANGLE TO SURFACE (RADIANS)
9 00000E-02	7	SODM	-MESH STAND-OFF DISTANCE (METERS)
3 0000	8	MOUNTF	-DISH MOUNTING FLAG. 0-APEX, 1-EDGE, 3-FREE
0 00000	9	REMOVF	-STRUT REMOVE FLAG 0-NO, 1-NEW SET, 9-REPEAT
15 000	10	NMODE	-NUMBER OF MODE SHAPES (0-NO SAP MODELS)
0 00000	11	XANACM	-X COORDINATE FOR ANGULAR ACCELERATION (METERS)
0 00000	12	YANACM	-Y COORDINATE FOR ANGULAR ACCELERATION (METERS)
0.00000	13	ZANACM	-Z COORDINATE FOR ANGULAR ACCELERATION (METERS)
1 0000	14	TUBTYP	-STRUT TYPE: 0-L/R, 1-EULER, 2-ISOG, 3-TRUSS
275.00	15	SLOR	-SURFACE STRUT LENGTH OVER RADIUS OF GYR RATIO
100 00	16	SDOT	-SURFACE STRUT DIAMETER OVER THICKNESS RATIO
1 30000E+11	17	SSYMM	-SURFACE STRUT YOUNGS MODULUS (NEWTONS/SQUARE METER)
4 00002E-02	18	SMINDM	-SURFACE STRUT MINIMUM DIAMETER (METERS)
1 00000E-03	19	SMINTM	-SURFACE STRUT MINIMUM THICKNESS (METERS
1003 0	20	SPCRM	-SURFACE STRUT EULER LOAD FOR DESIGN (NEWTONS)
0 75000	21	SHAR	-SURFACE STRUT HINGE AREA RATIO
0 10100	22	SHLR	-SURFACE STRUT HINGE LENGTH RATIO
1 10000E+11	23	SHMOD	-SURFACE STRUT HINGE MODULUS OR TRUSS LACE MODULUS
300.00	24	DLOR	-DIAGONAL STRUT LENGTH OVER RADIUS OF GYR RATIO
100 00	25	DDOT	-DIAGONAL STRUT DIAMETER OVER THICKNESS RATIO
1 30000E+11	26	DSYMM	-DIAGONAL STRUT YOUNGS MODULUS (NEWTONS/SQUARE METER)
2 00002E-02	27	DMINDM	-DIAGONAL STRUT MINIMUM DIAMETER (METERS)
1 00000E-03	28	DMINTM	-DIAGONAL STRUT MINIMUM THICKNESS (METERS)
1000 0	29	DPCRM	-DIAGONAL STRUT EULER LOAD FOR DESIGN (NEWTONS)
1 0000	30	DIFROD	-DIAGONAL STRUT PIN-ENDED (ROD) FLAG 0-BEAM, 1-ROD
1 38003E+10	31	DSSM	-DIAGONAL STRUT SHEAR MODULUS (N/M**2) USED IF DIFROD-0
1633 0	32	SRHO	-SURFACE STRUT DENSITY (KILOGRAMS/CUBIC METER)
4 54000E-03	33	SSBRW	-SURFACE STRUT BEARING REFERENCE WEIGHT (KILOGRAMS)
4 54000E-03	34	SEFRW	-SURFACE STRUT END FITTING REFERENCE WEIGHT (KILOGRAMS)
1633 0	35	SHLACE	-SURFACE STRUT HINGE OR LACE DENSITY (KILOGRAMS/METER**3)
1633 0	36	DRHOM	-DIAGONAL STRUT DENSITY (KILOGRAMS/CUBIC METER)
4 54000E-03	37	DSBRW	-DIAGONAL STRUT BEARING REFERENCE WEIGHT (KILOGRAMS)
4 54000E-03	38	DEFRW	-DIAGONAL STRUT END FITTING REFERENCE WEIGHT (KILOGRAMS)
1633 0	39	DLACE	-DIAGONAL STRUT TRUSS LACING DENSITY (KILOGRAMS/METER**3)
1 36000E-02	40	UWSTOM	-UNIT WEIGHT OF STAND-OFF (1 METERS LONG) (KILOGRAMS)
0 18140	41	SRWM	-SPIDER REFERENCE WEIGHT (KILOGRAMS)
7 00000E-02	42	UWMESH	-UNIT WEIGHT OF MESH (KILOGRAMS/SQUARE METER)
1 3000	43	UWMCSR	-MESH SYSTEM TO MESH ONLY WEIGHT RATIO
0 00000	44	WTCONT	-WEIGHT CONTINGENCY (ALL WEIGHT*(1+WTCONT))
0 00000	45	MESHBK	-MESH ON BACK FLAG (-1, ONLY IF FLAT) 0-FRONT

Table II. TTSS output summary.

GEODETIC TRUSS MASS PROPERTIES SUMMARY
TRUSS PARAMETERS 8 BAY 17 19 DEC AN 55 0 M (180 4 FT)ACROSS FLATS
440 L/RHO 51 D/T 1 500 F/D 0 0 % CONTINGENCY

COMPONENTS	UNIT WEIGHT KILOGRAMS	(POUNDS)	NUMBER REQUIRED	WEIGHT KILOGRAMS	(POUNDS)
STRUTS	0 17E+01	(0 37E+01)	420	0 70430E+03	(0 15530E+04)
UPPER SURFACE	0 21E+01	(0 46E+01)	156	0 32315E+03	(0 71255E+03)
TUBES	0 19E+01	(0 42E+01)	156	0 29645E+03	(0 65366E+03)
HINGES	0 13E+00	(0 28E+00)	156	0 19667E+02	(0 43365E+02)
END FITTINGS, PINS	0 13E-01	(0 29E-01)	312	0 41440E+01	(0 91374E+01)
BEARINGS	0 93E-02	(0 20E-01)	312	0 28975E+01	(0 63889E+01)
DIAGONALS	0 91E+00	(0 20E+01)	144	0 13072E+03	(0 28823E+03)
TUBES	0 88E+00	(0 19E+01)	144	0 12669E+03	(0 27936E+03)
END FITTINGS, PINS	0 76E-02	(0 17E-01)	288	0 21835E+01	(0 48147E+01)
BEARINGS	0 64E-02	(0.14E-01)	288	0.18405E+01	(0.40582E+01)
LOWER SURFACE	0 21E+01	(0 46E+01)	120	0 25043E+03	(0 55219E+03)
TUBES	0 19E+01	(0.42E+01)	120	0 22988E+03	(0 50689E+03)
HINGES	0 13E+00	(0 28E+00)	120	0 15128E+02	(0.33358E+02)
END FITTINGS, PINS	0 13E-01	(0 29E-01)	240	0 31877E+01	(0.70288E+01)
BEARINGS	0 93E-02	(0 20E-01)	240	0 22288E+01	(0 49145E+01)
SPIDER ASSEMBLY	0 31E+00	(0 68E+00)	109	0 33377E+02	(0.73596E+02)
UPPER SPIDER	0 30E+00	(0 66E+00)	61	0 18261E+02	(0 40266E+02)
LOWER SPIDER	0 30E+00	(0 66E+00)	48	0.14369E+02	(0 31684E+02)
STANDOFFS	0.12E-01	(0 27E-01)	61	0 74654E+00	(0.16461E+01)
NON-STRUCTURAL MASS				0.24007E+03	(0 52936E+03)
MESH				0.18467E+03	(0.40720E+03)
MESH CONTROL SYSTEM				0.55401E+02	(0 12216E+03)
CONTINGENCY				0.00000E+00	(0 00000E+00)
TOTAL WEIGHT				978	(2156)

X-C G CENTIMETERS (INCHES)	-0.38866E-02	(-1 5302E-03)
Y-C.G CENTIMETERS (INCHES)	-0.27701E-02	(-1 0906E-03)
Z-C G CENTIMETERS (INCHES)	0.84296E+02	(3 3187E+01)
IXX KILOGRAM METER SQ.(SLUG FT2)	0 21802E+06	(1 6068E+05)
IYY KILOGRAM METER SQ.(SLUG FT2)	0.21802E+06	(1.6068E+05)
IZZ KILOGRAM METER SQ.(SLUG FT2)	0.43377E+06	(3 1969E+05)
IXZ KILOGRAM METER SQ.(SLUG FT2)	-0 23645E-01	(-1 7426E-02)
IXY KILOGRAM METER SQ (SLUG FT2)	-0.18289E+00	(-1.3479E-01)
IZY KILOGRAM METER SQ (SLUG FT2)	-0.11812E-01	(-8.7052E-03)

MISCELLANEOUS GEOMETRY	UPPER SURFACE		DIAGONAL		LOWER SURFACE	
STRUT DIAMETER CM (IN)	5.20	(2.05)	3.58	(1 41)	5 20	(2 05)
STRUT THICKNESS CM (IN)	0 10000	(0 03937)	0 10000	(0 03937)	0 10000	(0 03937)
AVG STRUT LENGTH M (IN)	7	(313)	4	(188)	8	(316)
HINGE LENGTH CM (IN)	84.1	(33 1)	0 0	(0 0)	84 1	(33 1)
MAX STRUT LENGTH CM (IN)					808 6	(318 4)
MESH AREA SQ METERS (SQ YDS)					2638 8	(3156 1)
PACKAGE DIA CM (IN) A, B	315.2	(124 1)			137.2	(54 0)
PACKAGE HEIGHT CM (IN) A, B	536.7	(211 3)			1411 0	(555 5)
CRITICAL LOAD (PCR) NEWTONS (POUNDS)					1000 0	(224 8)

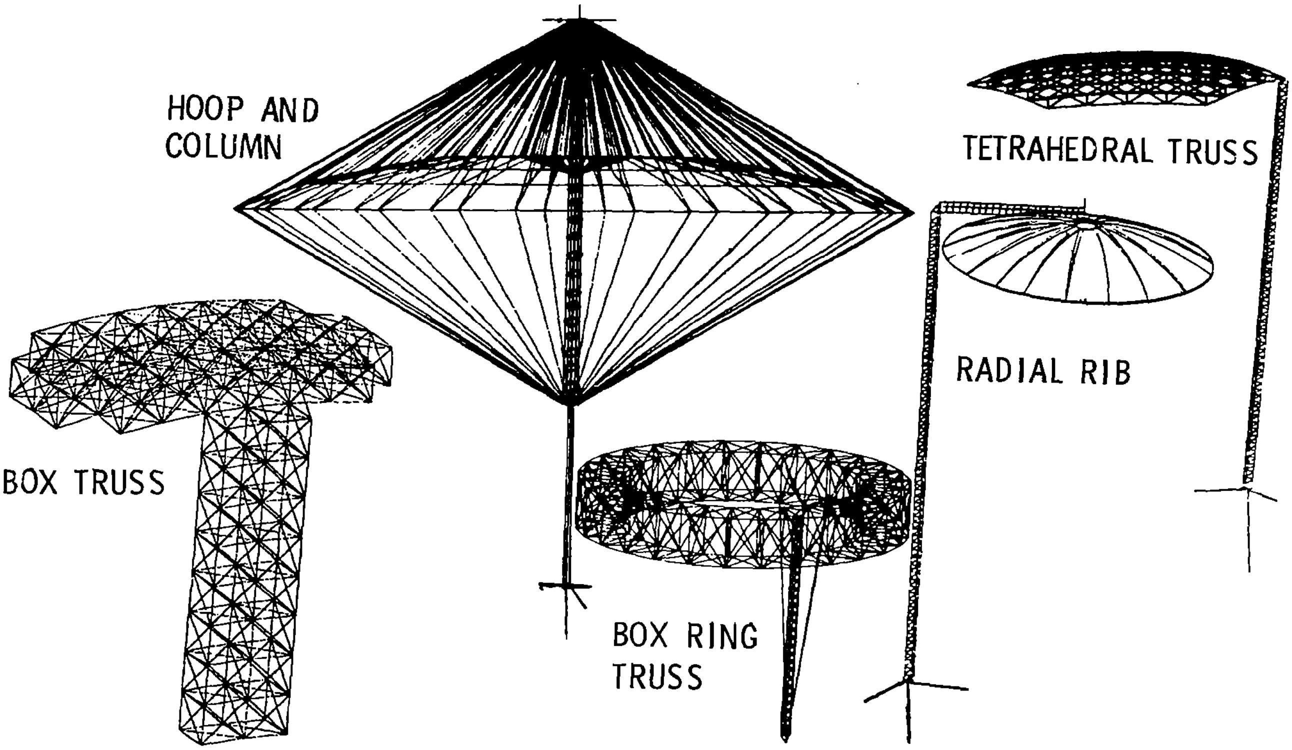


Figure 1. - Structural synthesizers.

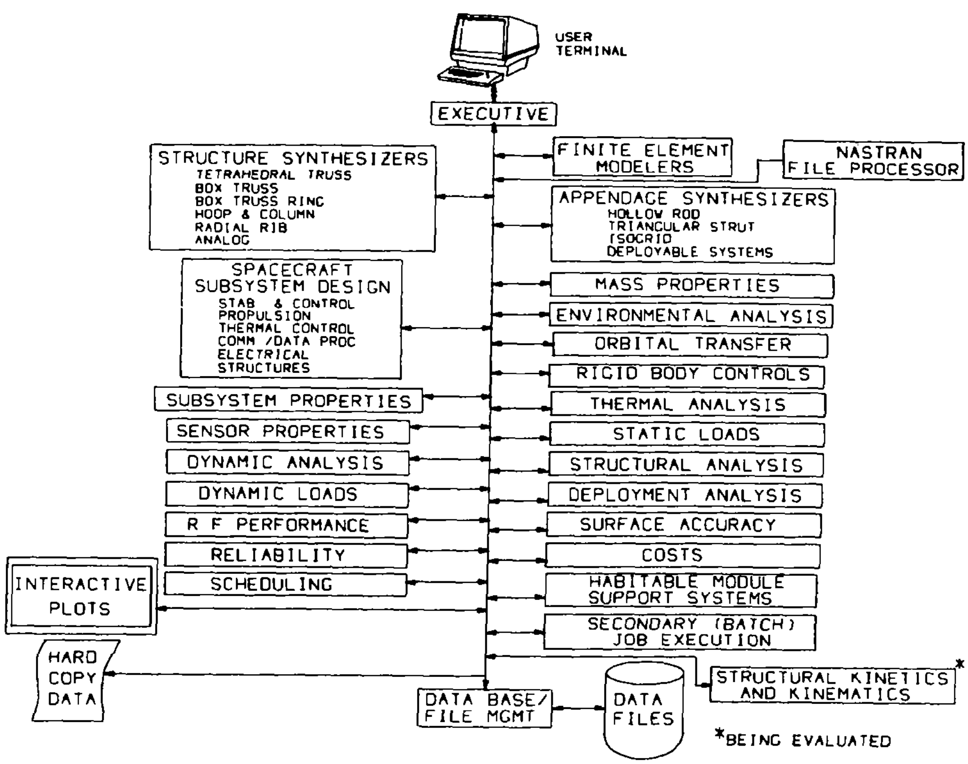


Figure 2. - IDEAS capabilities overview.

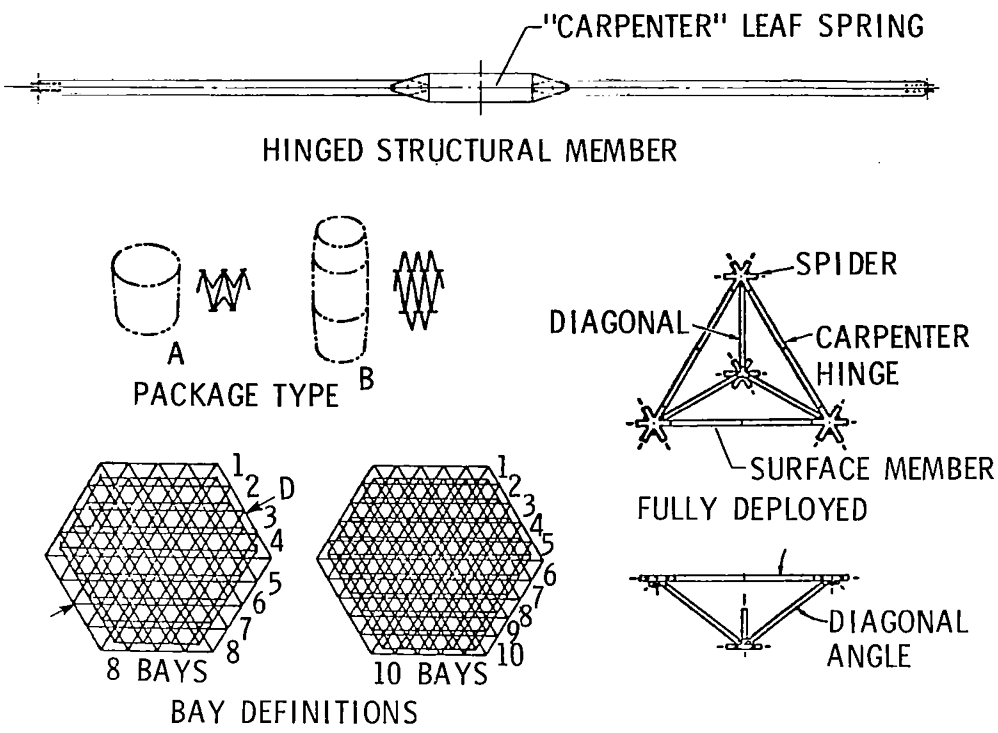


Figure 3(a). - Tetrahedral truss configuration definition.

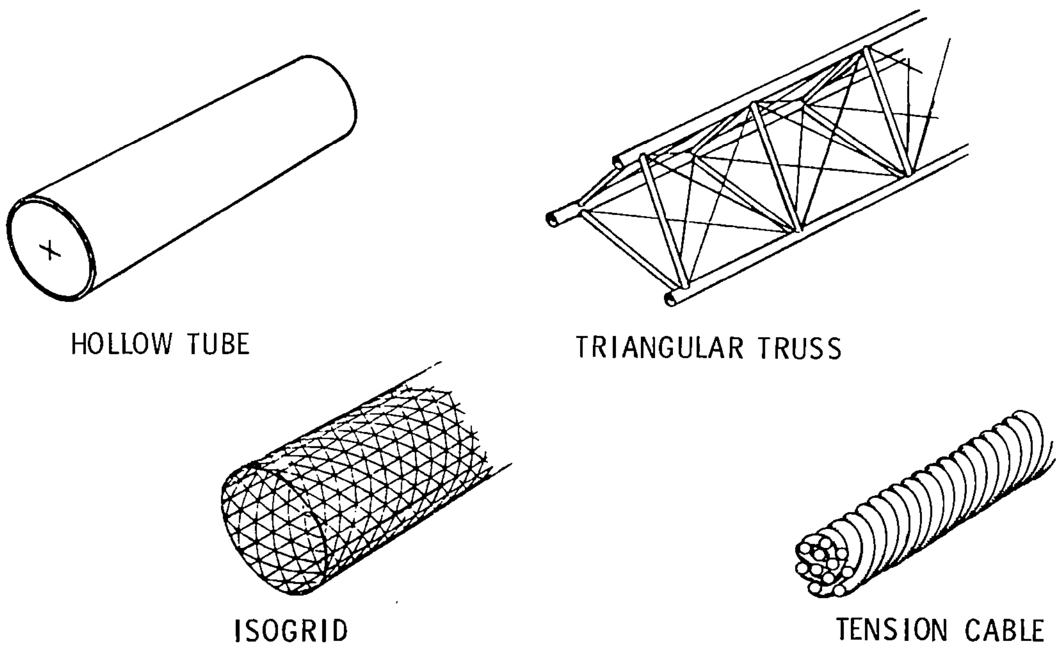


Figure 3(b). - Tetrahedral truss dish and appendages standard structural members.

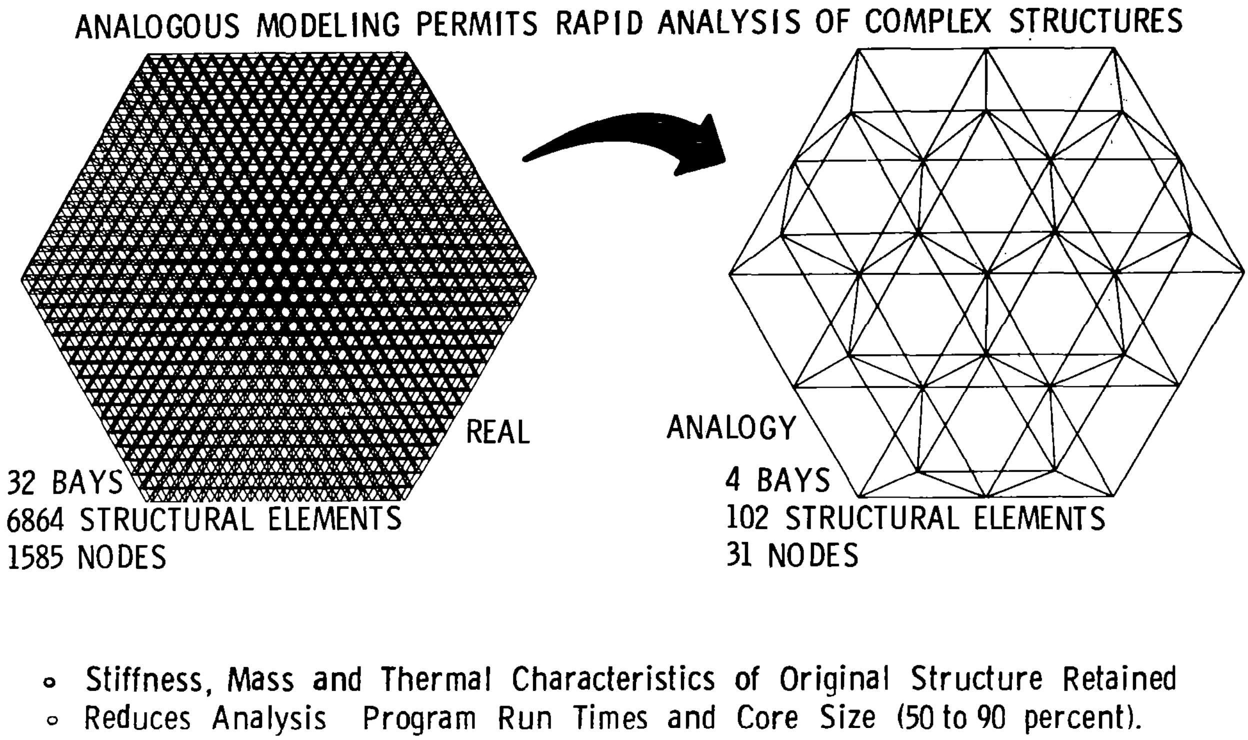


Figure 4. - Analogous modeling.

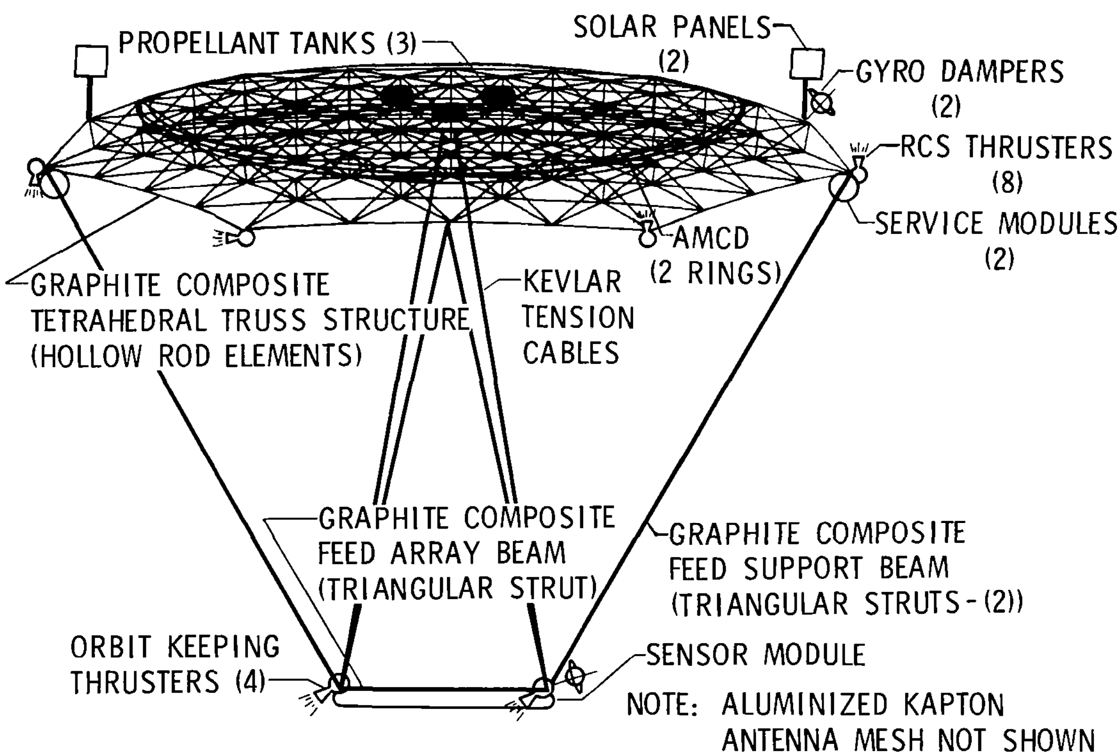


Figure 5. - Microwave radiometer spacecraft.

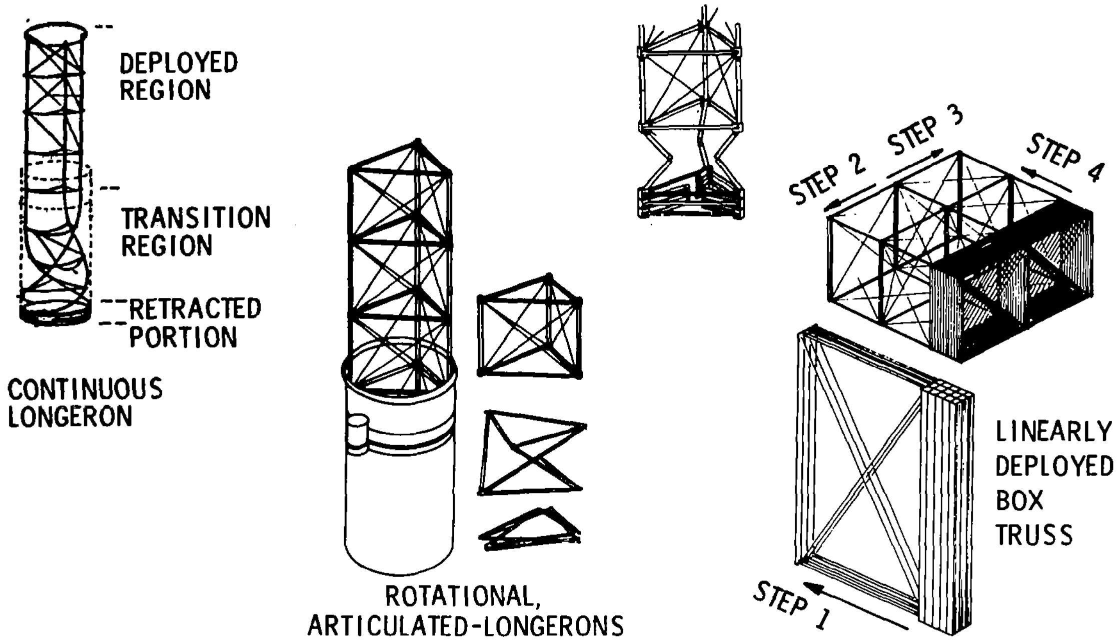


Figure 6. - Deployable structural appendage synthesizers.

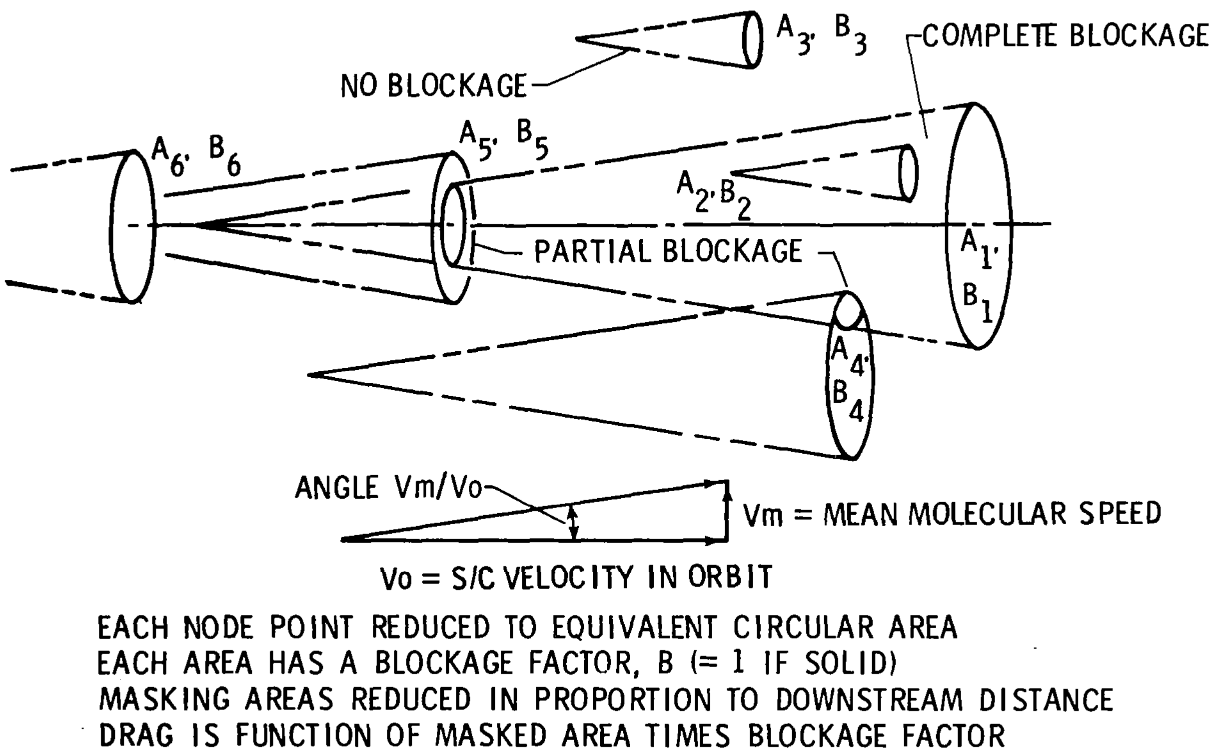


Figure 7. - Automated drag area computational approach.

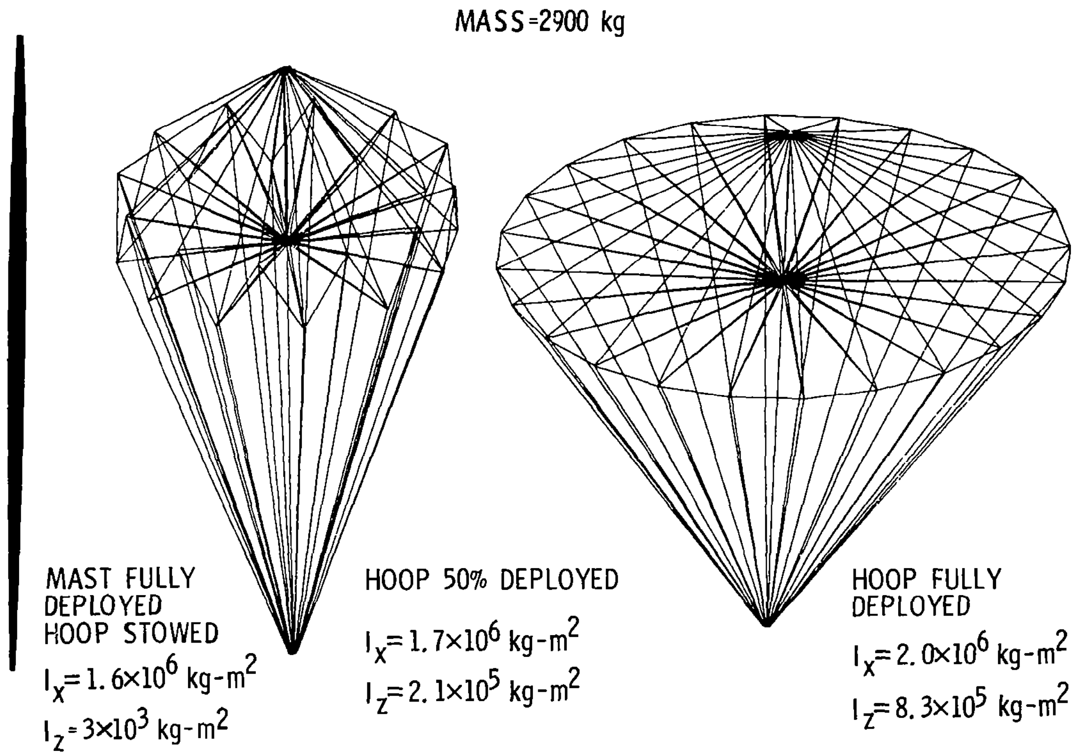


Figure 8. Inertia changes of a deploying hoop column antenna.

GEOMETRIC MODELING OF LARGE SPACE ANTENNA DEPLOYMENT

John Nazemetz
Oklahoma State University, Stillwater, Oklahoma

Karen Sage
Kentron International, Hampton, Virginia

Ray Gates
Computer Services Corporation, Hampton, Virginia

Darlene DeRyder
NASA Langley Research Center, Hampton, Virginia

The Systems and Experiments Branch of the Space Systems Division at Langley Research Center has been conducting several systems studies which involve the development, evaluation, and comparison of large space structures. One area of these studies concerns the deployment of large antennas in space. The purpose of this investigation was to create a dynamic visualization of large antenna deployment to be used as a precursor to further analysis of the structure as it is being deployed.

The configuration chosen for this investigation was a box-truss structure consisting of a large reflector 60 meters wide by 120 meters long with a 120-meter mast. The entire space structure is folded and compressed into a container 17.8 meters long by 3.75 meters in diameter for transportation to orbit by the Space Transportation System (STS).

A geometric model of this structure was created using an interactive solid modeling package developed by the Computer Sciences Corporation at LaRC. The building blocks of the structure are boxes and cylinders which are put together to form elements connected by joints and hinges. The elements, joints, and hinges make up the box-truss structure.

The geometric model was then passed to the MOVIE.BYU software package developed by Brigham Young University. The "utility" module of this program was used to manipulate the elements of the structure to represent 13 different stages of deployment. The "animate" command was then used to create a smooth sequence between stages. Each step in the sequence was recorded on magnetic tape and a short movie was produced.

1. Report No. NASA CP-2272	2. Government Accession No.	3. Recipient's Catalog No.
4. Title and Subtitle COMPUTER-AIDED GEOMETRY MODELING		5. Report Date March 1984
		6. Performing Organization Code 505-31-83-02
7. Author(s) John N. Shoosmith and Robert E. Fulton, compilers		8. Performing Organization Report No. L-15618
		10. Work Unit No.
9. Performing Organization Name and Address NASA Langley Research Center Hampton, VA 23665		11. Contract or Grant No.
		13. Type of Report and Period Covered Conference Publication
12. Sponsoring Agency Name and Address National Aeronautics and Space Administration Washington, DC 20546		14. Sponsoring Agency Code

15. Supplementary Notes

16. Abstract

The Symposium on Computer-Aided Geometry Modeling was held at NASA Langley Research Center on April 20-22, 1983. The purpose of the symposium was to communicate recent advances and to assess future directions and needs in the theory, methods, systems, and applications of computer-aided geometry modeling. This Conference Publication includes 47 papers: 8 were presented by invited speakers, 24 were selected as contributors, and 14 were presented at a poster session. The papers cover the following topics: (1) mathematical modeling of geometry, (2) solid geometry modeling, (3) the role of graphics in geometry modeling, (4) interactive modeling, (5) management of geometric data, (6) status of evolving national standards, (7) applications of geometry modeling, and (8) application of commercial systems.

17. Key Words (Suggested by Author(s))	18. Distribution Statement
Computer-aided design Computer-aided modeling IPAD Interactive computer modeling Geometry modeling Geometric data management	Unclassified - Unlimited Subject Category 59

19. Security Classif. (of this report)	20. Security Classif. (of this page)	21. No. of Pages	22. Price
Unclassified	Unclassified	406	A18

N-305

For sale by the National Technical Information Service, Springfield, Virginia 22161

☆U.S. GOVERNMENT PRINTING OFFICE: 1984 -739-010/ 62 REGION NO. 4

NASA-Langley, 1984